



- [Unisys Home](#)
- [Unisys Support Online](#)

ClearPath Enterprise Servers

OS 2200 Transaction Resource Adapter for the Java™ Platform User's Guide

Table Of Contents

OS 2200 Transaction Resource Adapter for the Java™ Platform User's Guide	1
What's New?	1
Documentation Updates	2
Java EE Connector Architecture Overview	2
Introduction to OS 2200 Transaction Resource Adapter for the Java™ Platform	4
Architecture	5
Java EE Connector Architecture 1.5 Requirements and Components	6
Services	9
Using OS 2200 Connector with WebLogic Server	10
Connector Installation and Configuration--WLS	10
Connector Administration--WLS	16
Using OS 2200 Connector with WebSphere Application Server	18
Connector Installation and Configuration--WebSphere Application Server	18
Connector Administration--WebSphere Application Server	24
Using OS 2200 Connector with JBoss Application Server	26
Connector Installation and Configuration--JBoss Application Server	26
Connector Administration--JBoss Application Server	35
Developing Your Solutions	38
Overview	38
Applications Using Communications Application Program Interface or Message Control Bank	39
Enterprise JavaBeans and Client Development	47
Client Development in a Nonmanaged Environment	54
Reference/Help Information	55

How to Use This HTML Help Document	55
Legal and Copyright Notices.....	59
Printing From This HTML Help Document	60
Index.....	63

OS 2200 Transaction Resource Adapter for the Java™ Platform User's Guide

What's New?

Change	Affected Topics
Support for additional TIP Session Control logon sequences.	<p>Provides user-defined alternates to the TIP Session Control standard logon dialogue. Customers might have sign-on requirements that include</p> <ul style="list-style-type: none">• Unique messages (before and after "Enter userid/password...")• Entry of accounts and project-ids in addition to userid and password <p>Three new config-property parameters (ClearanceLevel, AccountNumber, and ProjectId) define the default OS 2200 clearance level, account number/account number index, and project ID/project ID index used in the logon sequences. Another new config-property parameter (AuthenticationText) identifies the TIP session establishment message text.</p> <p>Configuring the Connector--WLS</p> <p>Configuring the Connector--WAS</p> <p>Configuring the Connector--JBoss Enterprise Application Platform</p>
Inbound Communication supports conversational mode where multiple EJB invocations can occur within the same connection.	<p>Provides a change to the inbound client API (header packet) with which the client can indicate OS 2200 TIP Connector to retain the inbound connection. OS 2200 TIP Connector requests more data from the client without closing the connection. This feature avoids the overhead of opening and closing connections if the client desires to make more than one Enterprise Java Bean (EJB) invocation.</p> <p>Inbound Communications</p>
Outbound communication examples for EJB 3.0 are provided for Oracle WebLogic Server and IBM WebSphere.	<p>EJB 3.0 sample programs and test instructions for WebLogic Server and WebSphere Application Server are included with the OS 2200 TIP Connector</p>

	<p>installation package. EJB 3.0 samples for outbound communication are provided.</p> <p>WebLogic Server EJB and Client Development</p> <p>WebSphere Application Server EJB and Client Development</p>
Inbound communication examples for EJB 3.0 are provided for Oracle WebLogic Server and IBM WebSphere.	<p>EJB 3.0 sample programs and test instructions for WebLogic Server and WebSphere Application Server are included with the OS 2200 TIP Connector installation package. EJB 3.0 samples for inbound communication are provided.</p> <p>EJB/POJO Development for Inbound Communications</p>
Support JBoss Enterprise Application Platform (EAP) 6.0	<p>Instructions to deploy OS 2200 TIP Connector with JBoss EAP 6.0 and EJB sample programs and test instructions.</p>

Documentation Updates

This document contains all the information that was available at the time of publication. Changes identified after release of this document are included in problem list entry (PLE) 18904101. To obtain a copy of the PLE, contact your Unisys representative or access the current PLE from the Unisys Product Support Web site:

<http://www.support.unisys.com/all/ple/18904101>

Note: If you are not logged into the Product Support site, you will be asked to do so.

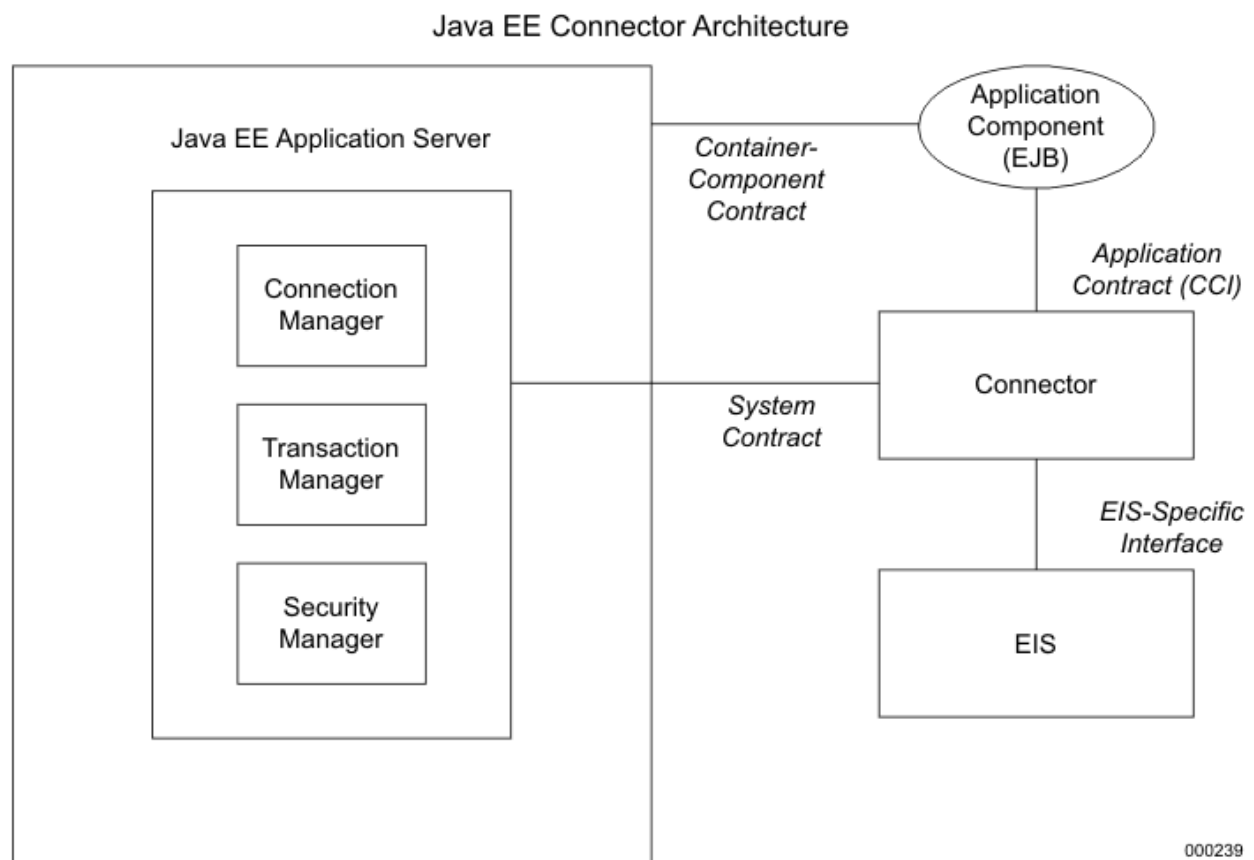
Java EE Connector Architecture Overview

The Java Enterprise Edition (Java EE) Connector Architecture specification defines a standard architecture for connecting Java EE platforms to Enterprise Information Systems (EIS). The Java EE Connector Architecture specification defines a set of scalable, secure, and transactional contracts that enable the integration of EISs (such as Unisys ClearPath Enterprise Servers) with Java EE Application Servers (such as Oracle WebLogic Server, IBM WebSphere Application Server, and JBoss® Enterprise Application Platform). Additionally, the specification also defines an Application Programming Interface (API) for access to EIS applications.

For more information on J2EE Connector Architecture (JCA), visit the [Java EE Connector Architecture Web site](#).

The Java EE Connector Architecture specification enables Unisys to provide a standard connector for ClearPath Enterprise Servers (OS 2200). A connector (also sometimes called a resource adapter) is a system-level software driver that is used by a Java application to connect to an existing system. The connector plugs into an Application Server and provides connectivity between the existing system, the Application Server, and the client application.

This connector allows Java EE Application Servers to perform connection management in the ClearPath Enterprise Server environment. Additionally, the connector allows Java components, such as Enterprise JavaBeans (EJB), to drive interactions with ClearPath Enterprise Server programs as illustrated in the following figure.



As shown in this figure, there are four major interfaces identified by the Java EE Connector Architecture specification:

- Container-Component Contract

This contract describes how an application component (EJB, Java servlet, or Java Server Page) accesses resources in a container managed by an Application Server.

- Application Contract API (CCI)

The client API is used by application components for EIS access. A ClearPath OS 2200 server is an example of an EIS. The Common Client Interface (CCI), as specified by the Java EE Connector Architecture specification, provides this access. Also, you can use a client API specific to the type of connector and its underlying EIS system to provide access to the EIS.

- System Contract

To achieve a standard system-level plug-in capability between Application Servers and EISs, the Java EE Connector Architecture defines a standard set of system-level contracts between an Application Server and EIS. The EIS side of these system-level contracts is implemented in the connector.

A connector is specific to an underlying EIS. It is a system-level software driver that is used by an Application Server or an application client to connect to an EIS. A connector plugs into an Application Server. The connector and Application Server collaborate to provide the underlying mechanisms--transactions, security, and connection pooling--to interoperate with the EIS.

A connector is used within the address space of the Application Server.

- EIS-Specific Interface

Outbound communications: The connector defines a communication protocol to provide access to the EIS system. The connector converts the Java EE application component requests into EIS specific requests. The connector allows components to work together that could not otherwise because of incompatible interfaces.

Inbound communications: EIS inbound clients use an EIS-specific communication protocol to provide access to Enterprise Java Beans (EJBs). The connector provides a mechanism for EIS client applications to invoke EJBs inside an application server.

[Return to Top](#)

Introduction to OS 2200 Transaction Resource Adapter for the Java™ Platform

- [Architecture](#)
- [Java EE Connector Architecture 1.5 Requirements and Components](#)
- [Services](#)

The OS 2200 Transaction Resource Adapter for the Java™ Platform (OS 2200 Connector or TIP RA or J2EE-CON-OS2200) provides access from application components executing under Oracle WebLogic Server (WLS), IBM WebSphere Application Server (WAS), and JBoss® Enterprise Application Platform to ClearPath OS 2200 TIP, HVTIP, DPS, and batch programs. OS 2200 Connector clients and technologies include Enterprise JavaBeans (EJB), Java Server Pages (JSP), Java servlets, and other J2EE technologies. For more information on Java EE connectors, see [Java EE Connector Architecture Overview](#).

OS 2200 Connector enables you to make new or existing applications running in the OS 2200 operating environments available to clients in Java platform networks outside the legacy systems.

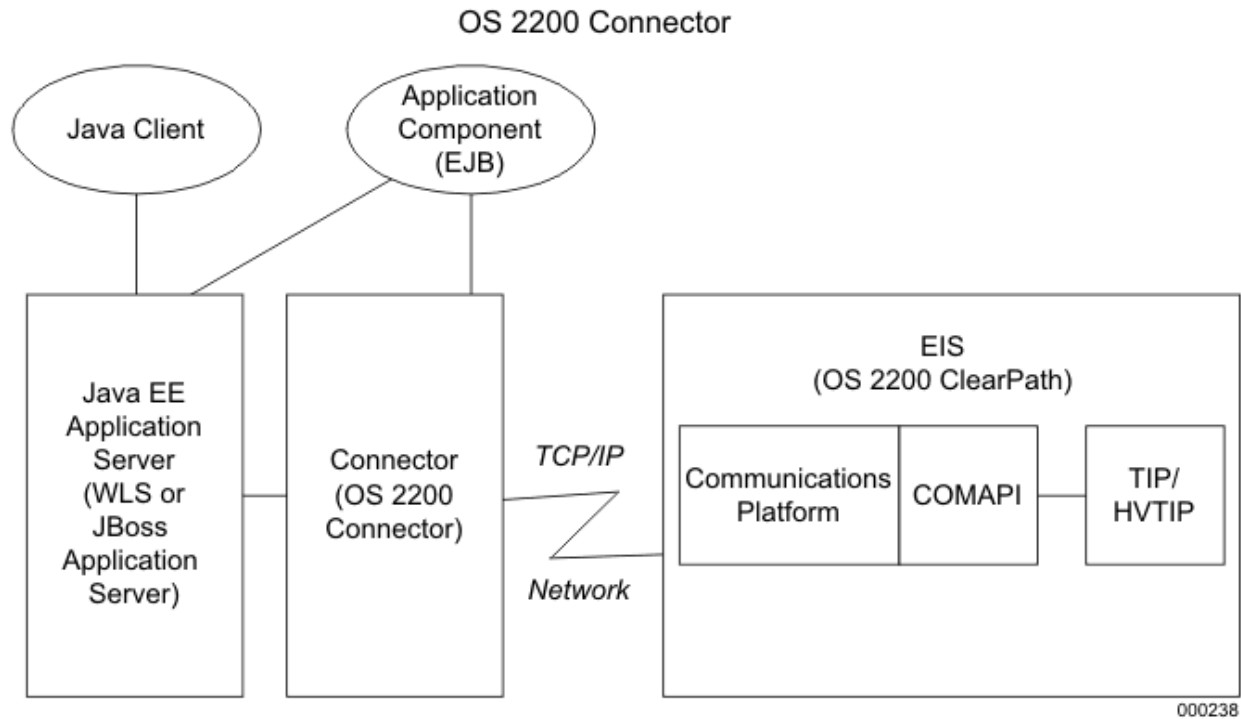
Audience

The primary audiences for this online guide are application component developers and system administrators who are responsible for preparing, installing, and deploying OS 2200 Connector along with development tasks associated with using the product. Developers should also refer to the API documentation located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Architecture

OS 2200 Connector is a Java EE Connector Architecture component that is deployed with WLS, WAS, or JBoss Enterprise Application Platform on a Java platform. Once deployed and active, OS 2200 Connector acquires a configured set of underlying physical TCP/IP socket connections, which are pooled and managed by the Application Servers. OS 2200 Connector accepts application component connection requests for OS 2200 and subsequent interactions for OS 2200 programs through the connection. When OS 2200 Connector receives a connection request, it matches that request with an underlying socket connection, which may or may not already be pooled by the Application Server. If pooled, the request is granted; otherwise, a new physical connection, which matches the application component connection request, is acquired and added to the Application Server connection pool. Once a component connection request is granted, the component can interact with OS 2200 TIP/HVTIP, DPS, or batch programs using a request/response paradigm.

The following figure shows the basic OS 2200 Connector architecture, where OS 2200 Connector acts as a driver to access OS 2200 Communications Application Program Interface applications. The application component can be an EJB, JSP, or Java servlet running under the Application Server container.



[Return to Top](#)

Java EE Connector Architecture 1.5 Requirements and Components

- [Connection Management](#)
- [Common Client Interface](#)
- [Transaction Management](#)
- [Managed Environment](#)
- [Security Management](#)
- [Nonmanaged Environment](#)
- [Inbound Communication](#)

OS 2200 Connector provides a subset of the Java EE connector capabilities specified by Java EE Connector Architecture version 1.5. Future versions of OS 2200 Connector will add more capabilities and lift more restrictions.

This section outlines requirements and components of Java EE Connector Architecture 1.5 and how OS 2200 Connector satisfies and implements the requirements and components.

Connection Management

Connection management specifies a contract that allows an Application Server to pool connections to an underlying EIS, and lets application components connect to an EIS. The connection management contract places the low-level connection routines (for example, socket creation) on the connector and the connection management routines (for example, connection pooling) on the Application Server.

OS 2200 Connector provides a Connection Management contract with WLS, WAS, or JBoss Enterprise Application Platform, which pools TCP/IP socket connections to the OS 2200

Communications Application Program Interface. This allows OS 2200 Connector to support a large number of physical connections for clients requiring access to OS 2200 programs. OS 2200 Connector does not currently propagate transaction context to the OS 2200 environment.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Transaction Management

The transaction management contract builds on the connection management contract. It specifies the types of transactions supported by Java EE Connector Architecture, and describes how an Application Server and a connector coordinate their efforts to include an OS 2200 program in transactions managed by an Application Server or demarcated by an application component.

OS 2200 Connector does not currently support a transaction management contract.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Security Management

The security management contract of Java EE Connector Architecture extends the connection management contract by allowing an application component or an Application Server to add security specific details. The security management contract supports both container-managed sign-on and component-managed sign-on.

OS 2200 Connector provides a Security Management contract with WLS, WAS, or JBoss Enterprise Application Platform. OS 2200 user-ids and passwords are used to authenticate application components or application servers with OS 2200 TIP Session Control. In the component-managed case, the application component specifies the OS 2200 user-id and password using the Common Client Interface. In the container-managed case, the application server specifies the OS 2200 user-id and password in a container provided login module. The application server propagates the container-managed security context to OS 2200 Connector, and OS 2200 Connector authenticates the user-id/password with OS 2200 TIP Session Control.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Common Client Interface

The CCI defines a standard client API for application components. The CCI enables application components and Enterprise Application Integration (EAI) frameworks to drive interactions across heterogeneous EISs using a common client API. The target users of the CCI are enterprise tool vendors and EAI vendors. Application components themselves may also write to the API, but the CCI is a low-level API. The specification recommends that the

CCI be the basis for richer functionality provided by the tool vendors, rather than being an application-level programming interface used by most application developers.

OS 2200 Connector provides a base-level CCI. The API constituents are a set of Java classes that allow an application component to manually create and configure data objects to drive an interaction with an OS 2200 TIP or HVTIP user program. For a list of the classes, see OS 2200 Connector API documentation, located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Managed Environment

OS 2200 Connector supports a managed environment, which defines an operational environment for a J2EE-based, multi-tier, Web-enabled application that accesses EISs. The application consists of one or more application components (for example, EJBs, JSPs, or Java servlets) that are deployed on containers such as WLS, WAS, or JBoss Enterprise Application Platform. These containers can be one of the following:

- Web containers that host JSP, servlets, and static HTML pages
- EJB containers that host EJB components
- Application client containers that host stand alone application clients

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Nonmanaged Environment

OS 2200 Connector also supports a nonmanaged environment, which defines an operational environment for a two-tier application. An application client directly uses a connector to access the EIS, which defines the second tier for a two-tier application. A connector, in this case, exposes its low-level transactions and security APIs to its clients. An application component is responsible for managing security and transactions, and relies on the connection pooling performed by the connector.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Inbound Communication

OS 2200 Connector also supports inbound communication that allows EIS client applications written in C, COBOL, or Java to invoke Enterprise JavaBeans (EJBs) that are deployed in application servers such as JBoss Enterprise Application Platform, Oracle WLS, or WAS. Both EJB 2.0 and EJB 3.0 target EJBs are supported. Spring POJO (Plain Old Java Object) Beans are also supported as compliant targets.

TCP socket connections or MCB sessions can be used to transfer data between the client applications and OS 2200 Connector compliant EJBs and Spring POJO Beans.

Client applications written in C or COBOL use the Communications Application Program Interface to make their socket connections. Java provides its own set of socket routines that are used to connect with OS 2200 Connector.

Client applications written in C or COBOL may also use the Message Control Bank (MCB) to interact with OS 2200 Connector compliant EJBs and Spring POJO Beans.

[Return to Top](#)

[Return to Top of Java EE Connector Architecture 1.5 Requirements and Components](#)

Services

OS 2200 Connector provides the following services:

- Data handling

OS 2200 Connector does not provide for data conversion between the client environment and the legacy environment. It supports only data transfers where no data conversion occurs (octet data), leaving data translation issues entirely to the clients and services. OS 2200 Connector can pass Record (or VIEW) names of the data requirements of the legacy system. The Record name is passed as part of the data record object (OS2200Record). The Record name can be used to identify the structure of the data passed between the clients and services.

- Server handling and communication

OS 2200 Connector communicates with the host application as though it were the client, ensuring that all I/O is sent and received correctly from the legacy application perspective.

[Return to Top](#)

Related Topics

- [Java EE Connector Architecture Overview](#)
- [Programming WebLogic Resource Adapters Web page](#)
- [WebSphere Application Server, Version 6.1 Information Center Web site](#)
- [JBoss Enterprise Application Platform Product Documentation](#)

Using OS 2200 Connector with WebLogic Server

Connector Installation and Configuration--WLS

Installing the Connector--WLS

Insert the OS 2200 Connector CD-ROM into the CD-ROM drive. Open setup.exe in the Win32 folder. Unisys recommends the default installation folder be named C:\Unisys\J2EE_Connector\OS2200 to coincide with the sample client and package generation scripts.

After installation, the OS 2200 Connector must be deployed as a component in Oracle WebLogic Server 10.3 before any application components can use it. For information on OS 2200 Connector configuration and deployment, see [Configuring the Connector--WLS](#).

Related Topics

- [Configuring the Connector--WLS](#)
- [Using the WebLogic Server Administration Console](#)

Configuring the Connector--WLS

- [ra.xml](#)
- [weblogic-ra.xml](#)

This section tells how to configure OS 2200 Connector for deployment to Oracle WebLogic Server. For additional information on connector configuration in WLS, see the [Creating and Configuring Resource Adapters](#) web page.

OS 2200 Connector must be deployed on a Oracle WebLogic Server 10.3. To learn about installing and administering WLS, and to learn about deploying components within WLS, see the [Oracle WebLogic Server Release 10.3 Documentation](#) web page.

For detailed information on developing and administering the Resource Adapters in WLS, see the [Programming WebLogic Resource Adapters](#) web page.

ra.xml

If you do not have an ra.xml deployment descriptor file, you must manually create or edit an existing one to set the necessary deployment properties for OS 2200 Connector. You can use a text editor to edit the properties. For information on the structure of the ra.xml deployment descriptor, see the [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#) web site.

OS 2200 Connector provides a sample ra.xml deployment descriptor file with its installation. You must edit the config-property sections of the ra.xml file to use your own OS 2200 Connector configuration properties.

- `config-property-name: ServerName`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the host name or IP address of the OS 2200 ClearPath server. If you are configuring OS 2200 Connector for use in a multi-host high availability environment with XTC (eXtended Transaction Capacity), this property will contain a semi-colon delimited list of configured host names or IP addresses.
- `config-property-name: PortNumber`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the port number of the OS 2200 Communications Application Program Interface listener defined in the CITA configuration. OS 2200 Connector initially opens a configuration-dependent number of connections (specified by the initial-capacity property in `weblogic-ra.xml`) that are initially managed and pooled by WLS upon OS 2200 Connector deployment.
- `config-property-name: ConnectionTO`
`config-property-type: java.lang.Integer`
`config-property-value:` This property specifies the connection timeout in milliseconds. This is the amount of time that an outbound client waits to receive a response or more data from an OS 2200 transaction. A value of zero disables this feature. Default = 0.
- `Config-property-name: UseSecureConnection`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the Secure Socket Layer (SSL) indicator for outbound communications. If true, OS 2200 Connector initiates an SSL handshake with OS 2200 Communications Platform (CPComm). An encryption method and a unique session key are established between the OS 2200 Connector client and the OS 2200 ClearPath server. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for outbound communications, the PortNumber config-property parameter must match a CPComm configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A trusted certificate file that matches the certificate on the OS 2200 ClearPath Server must be imported to a "truststore" on the system containing the WebLogic Server Java Virtual Machine (JVM). In the WLS startup script, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore;" that is, use the java option `-Djavax.net.ssl.trustStore=C:\...\my.truststore`. For more information on creating and managing digital certificates and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation.

- `Config-property-name: UserName`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 user-id to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `Config-property-name: Password`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 password to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.

- `config-property-name: ClearanceLevel`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 clearance-level to be used for connection authentication with TIP Session Control on OS 2200. Default = `""`. Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: ProjectId`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 project-id or project-id-index to be used for connection authentication with TIP Session Control on OS 2200. Default = `""`. Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AccountNumber`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 account-number or account-number-index to be used for connection authentication with TIP Session Control on OS 2200. Default = `""`. Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AuthenticationText`
`config-property-type: java.lang.String`
`config-property-value:` This property specifies the last response message text from TIP Session Control on OS 2200 to indicate when a session has been successfully established. Default = `"Previous session was:"`. Use `""` if TIP Session Control is not configured on OS 2200.
- `config-property-name: LogFilename`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the directory path/file name of the OS 2200 Connector log file, where event and diagnostic trace information is recorded. If the path is a relative path name, the log file is in the Server Root Directory for your WLS instance. See the [Understanding WebLogic Domain Configuration](#) web page.
- `config-property-name: LoggingEnabled`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the event logging indicator. If true, event information is recorded in the OS 2200 Connector log file. The event log records primary events and errors in the usage of OS 2200 Connector.
- `config-property-name: Debug`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the OS 2200 Connector diagnostic trace indicator. If true, OS 2200 diagnostic trace entries are logged to the OS 2200 Connector log file.
- `config-property-name: MaximumCharacters`
`config-property-type: java.lang.Integer`
`config-property-value:` This property contains the maximum character string length for input and output data transported between OS 2200 Connector and the TIP/HVTIP transaction. Default = 512.
- `config-property-name: InPort`
`config-property-type: java.lang.Integer`

`config-property-value`: This property contains the IP address that OS 2200 Connector listens on for inbound requests. Default = 2498.

- `config-property-name`: `UseInboundSecureConnection`
`config-property-type`: `java.lang.Boolean`
`config-property-value`: This property contains the Secure Socket Layer (SSL) indicator for inbound communications. If true, the OS 2200 client program initiates an SSL handshake with OS 2200 Connector. An encryption method and a unique session key are established between the OS 2200 client program and OS 2200 Connector. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for inbound communications, the `InPort` `config-property` parameter must match a `CPComm` configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A private key and associated trusted digital certificate must be generated on the system containing the WebLogic Server Java Virtual Machine (JVM). The key must be imported into the Java system "keystore" and the certificate must be imported into the Java system "truststore." In the WLS startup script, the `javax.net.ssl.keyStore` Java system property must be set to the location of the "keystore"; that is, use the java option - `Djavax.net.ssl.keyStore=C:\... \my.keystore`. Also, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore." For more information on creating and managing keys, digital certificates, keystores and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation. Finally, the trusted certificate file must also be imported into the *trusted-certificates-file* declared in the OS 2200 ClearPath Server `CPComm` configuration.

- `config-property-name`: `RequestSocketTO`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that OS 2200 Connector waits to receive more data from an inbound client request. A value of zero disables this feature. Default = 10000.
- `config-property-name`: `ServerSocketTO`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that OS 2200 Connector waits for a new inbound client request. A value of zero disables this feature. Default = 0.

The `ServerName`, `PortNumber`, `ConnectionTO`, `UseSecureConnection`, `UserName`, `Password`, `ClearanceLevel`, `ProjectId`, and `AccountNumber` parameters define the default outbound connection attributes for OS 2200 Connector. To override these attributes in your application component, create an `OS2200ConnectionSpec` object with the new parameter values. Use the `OS2200ConnectionSpec` object to obtain the `OS2200Connection` object. For more information, see OS 2200 Connector API documentation, located at (assuming default installation path) `C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html`. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Unisys recommends that the remainder of the properties in `ra.xml` remain unchanged. Modify those properties only if necessary to tailor to your installation needs.

[Return to Top](#)

weblogic-ra.xml

In addition to supporting features of the standard connector configuration ra.xml deployment descriptor file, Oracle WebLogic Server defines an additional deployment descriptor file, the weblogic-ra.xml file. This file contains properties that are specific to configuring and deploying connectors in WLS. This functionality adds WLS-specific deployment descriptors to the deployable archive. As is, the basic RAR or deployment directory cannot be deployed to Oracle WebLogic Server. You must first create and configure WLS-specific deployment properties in the weblogic-ra.xml deployment descriptor and add that file to the deployment.

Note: For more information on setting the properties in weblogic-ra.xml, see the weblogic-ra.xml schema in the [weblogic-ra.xml Schema](#) web page.

OS 2200 Connector provides a sample weblogic-ra.xml deployment descriptor file with its installation.

Edit the following properties of the weblogic-ra.xml file to suit your needs:

- jndi-name property to configure your own connection factory JNDI name.
- pool-params properties, for connection pooling and management.
- security properties, to configure container-managed sign-on.

[Return to Top](#)

Related Topics

- [Generating the Connector Archive--WLS](#)
- [Using the WebLogic Server Administration Console](#)
- [Creating and Configuring Resource Adapters](#)
- [Oracle WebLogic Server Release 10.3 Documentation](#)
- [Programming WebLogic Resource Adapters](#)
- [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#)
- [weblogic-ra.xml Schema](#)
- [Understanding WebLogic Domain Configuration](#)

Generating the Connector Archive--WLS

To create the OS 2200 Connector archive (RAR), follow these steps:

1. Create an empty staging directory.
2. Place the deployment descriptors (ra.xml and weblogic-ra.xml) in a subdirectory called META-INF.
3. Create the connector archive by executing a jar command like the following in the staging directory:

```
jar cvf OS2200.rar *
```

4. Verify the connector archive was built correctly by executing a jar command like the following:

```
jar -tf OS2200.rar
```

The OS2200.rar file structure should look like the following:

```
META-INF/
META-INF/MANIFEST.MF
META-INF/ra.xml
META-INF/weblogic-ra.xml
```

5. Copy the jar containing the OS 2200 Connector Java classes (OS2200.jar) to the *lib* directory of your WebLogic Server domain. For example:
C:\Oracle\Middleware\wlserver_10.3\samples\domains\wl_server\lib.
6. Deploy the RAR on WebLogic Server or include it in an enterprise archive (EAR) to be deployed as part of an enterprise application (See [Deploying the Connector--WLS](#)).

The OS 2200 Connector installation provides sample command scripts (setEnv.cmd and buildrar.cmd) that you can use, with site-specific modifications, to build the RAR.

Related Topics

- [Deploying the Connector--WLS](#)
- [Using the WebLogic Server Administration Console](#)

Deploying the Connector--WLS

To deploy OS 2200 Connector on WLS 10.3, follow these steps:

Note: WLS must be running and active.

1. If not already created, create a server using the Domain Configuration Wizard. For information regarding domain configuration, see [Creating WebLogic Domains Using the Configuration Wizard](#) web page.
2. Start the server.
3. Open and log on to the WebLogic Server Administration Console. Open `http://ip-address:7001/console` in your browser window, where *ip-address* is the WLS execution host location.
4. Select **Deployments** in the **Domain Structure** window.
5. Click the **Install** button under **Deployments**.
6. Browse to your OS 2200 RAR or EAR file and select it; then click **Next**.
7. In the **Install Application Assistant** page, select **Install this deployment as an application**, and click **Next**.
8. In the **Install Application Assistant** page, type a name for the connector module, such as **OS 2200 TIP Connector**. Select **Use the defaults defined by the deployment's targets** and click **Next**.
9. In the **Install Application Assistant** page, select **No, I will review the configuration later**, and click **Finish**.
10. Click **Deployments** in the **Domain Structure** window. The **Summary of Deployments** page now shows the connector module marked in the **Active** state.

Once deployed with this procedure, the connector is automatically redeployed when WLS restarts.

For further information regarding enterprise application deployment using the WLS Administration Console, see the [Deploying WebLogic Enterprise Applications](#) web page.

[Return to Top](#)

Related Topics

- [Generating the Connector Archive--WLS](#)
- [Editing the Deployment Descriptors--WLS](#)
- [Using the WebLogic Server Administration Console](#)
- [Creating WebLogic Domains Using the Configuration Wizard](#)
- [Deploying WebLogic Enterprise Applications](#)

Connector Administration--WLS

Using the WebLogic Server Administration Console

To administer OS 2200 Connector, use the WebLogic Server System Administration Console.

For general information about using the WebLogic Server Administration Console, see the [WebLogic Server Documentation--Administration Console Online Help](#) web page.

To open the WAS Administration Console, open a web browser window and browse to:

`http://ip-address: 7001/console`

where *ip-address* is the WLS execution host location.

For specific information about administering J2EE Connector Architecture connectors in the WebLogic Server Administration Console, see [WebLogic Server Documentation--Administration Console Online Help--Configure Resource Adapters](#) web page.

Related Topics

- [WebLogic Server Documentation--Administration Console Online Help](#)
- [WebLogic Server Documentation--Administration Console Online Help--Configure Resource Adapters](#)

Editing the Deployment Descriptors--WLS

You can edit the following connector deployment descriptors using a text editor or XML editor:

- ra.xml
- weblogic-ra.xml

You can edit the following OS 2200 Connector specific config properties of ra.xml:

- Debug
- LogFilename

- LoggingEnabled
- PortNumber
- ServerName
- ConnectionTO
- UseSecureConnection
- UserName
- Password
- ClearanceLevel
- ProjectId
- AccountNumber
- AuthenticationText
- MaximumCharacters
- InPort
- UseInboundSecureConnection
- RequestSocketTO
- ServerSocketTO

You can edit the following properties of weblogic-ra.xml:

- pool-params properties
- jndi-name
- security properties

After you save the edited changes, undeploy OS 2200 Connector and rebuild the RAR archive with the new .xml files. Then you can redeploy OS 2200 Connector.

[Return to Top](#)

Related Topics

- [Deploying the Connector](#)
- [Using the WebLogic Server Administration Console](#)

Event Log and Diagnostic Trace Log--WLS

The J2EE Connector Architecture provides for a logging mechanism by which connectors can log arbitrary events to a predetermined file location. OS 2200 Connector uses this log mechanism to record primary events, warnings, errors, and diagnostic trace information during the execution of OS 2200 Connector. The log file is identified by the LogFilename property of the ra.xml deployment descriptor. Both the event log and the diagnostic trace log can be enabled and disabled independently using ra.xml properties.

OS 2200 Connector creates a log file set with a maximum of five cycles. The size limit for each log file in the set is 5MB. When the active file in the set reaches the 5MB size limit, this file is closed and the next file in the set is opened. When the last and fifth file in the set reaches the size limit, the oldest file in the set is purged.

OS 2200 Connector identifies the files in the set by appending a number, starting with 0, to the base file name. For example, if the base file name is OS2200Connector.log and the set consists of five files, the files are named OS2200Connector.log.0, OS2200Connector.log.1, OS2200Connector.log.2, OS2200Connector.log.3, and OS2200Connector.log.4. The OS2200Connector.log.0 file always contains the most recent cycle written. When OS 2200 Connector restarts, it appends log entries to the most recent log file cycle that was in use when OS 2200 Connector was stopped.

To view the log file, open it with a text editor.

Related Topics

- [Using the WebLogic Server Administration Console](#)
- [Configuring the Connector--WLS](#)

Using OS 2200 Connector with WebSphere Application Server

Connector Installation and Configuration--WebSphere Application Server

Installing the Connector--WAS

Insert the OS 2200 Connector CD-ROM into the CD-ROM drive. Open setup.exe in the Win32 folder. Unisys recommends the default installation folder be named C:\Unisys\J2EE_Connector\OS2200 to coincide with the sample client and package generation scripts.

After installation, the OS 2200 Connector must be deployed as a component in IBM WebSphere Application Server 8.0 before any application components can use it. For information on OS 2200 Connector configuration and deployment, see [Configuring the Connector--WAS](#).

Related Topics

- [Configuring the Connector--WAS](#)
- [Using the WebSphere Application Server Console](#)

Configuring the Connector--WAS

- [ra.xml](#)

This section introduces and discusses how to configure OS 2200 Connector for deployment to IBM WebSphere Application Server. For additional information on connector configuration in WAS, see the [WebSphere Application Server, Version 8.0 Documentation - Configuring Resource Adapters](#) web page.

OS 2200 Connector must be deployed on a IBM WebSphere Application Server 6.0. To learn about installing and administering WAS, and to learn about deploying components within WAS, see the [WebSphere Application Server, Version 8.0 Information Center](#) web site.

ra.xml

If you do not have an ra.xml deployment descriptor file, you must manually create or edit an existing one to set the necessary deployment properties for OS 2200 Connector. You can use a text editor to edit the properties. For information on the structure of the ra.xml deployment descriptor, see the [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#) web site.

OS 2200 Connector provides a sample ra.xml deployment descriptor file with its installation. You must edit the config-property sections of the ra.xml file to use your own OS 2200 Connector configuration properties.

- `config-property-name: ServerName`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the host name or IP address of the OS 2200 ClearPath server. If you are configuring OS 2200 Connector for use in a multi-host high availability environment with XTC (eXtended Transaction Capacity), this property will contain a semi-colon delimited list of configured host names or IP addresses.
- `config-property-name: PortNumber`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the port number of the OS 2200 Communications Application Program Interface listener defined in the CITA configuration. OS 2200 Connector opens a configuration dependent number of connections that are initially managed and pooled by WAS upon OS 2200 Connector deployment.
- `config-property-name: ConnectionTO`
`config-property-type: java.lang.Integer`
`config-property-value:` This property specifies the connection timeout in milliseconds. This is the amount of time that an outbound client waits to receive a response or more data from an OS 2200 transaction. A value of zero disables this feature. Default = 0.
- `config-property-name: UseSecureConnection`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the Secure Socket Layer (SSL) indicator for outbound communications. If true, OS 2200 Connector initiates an SSL handshake with OS 2200 Communications Platform (CPComm). An encryption method and a unique session key are established between the OS 2200 Connector client and the OS 2200 ClearPath server. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for outbound communications, the PortNumber config-property parameter must match a CPComm configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A trusted certificate file that matches the certificate on the OS 2200 ClearPath Server must be imported to a "truststore" on the system containing the WebSphere Application Server Java Virtual Machine (JVM). In the WAS startup script, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore"; that is, use the java option - `Djavax.net.ssl.trustStore=C:\...\my.truststore`. For more information on creating

and managing digital certificates and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation.

- `config-property-name: UserName`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 user-id to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: Password`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 password to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: ClearanceLevel`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 clearance-level to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: ProjectId`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 project-id or project-id-index to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AccountNumber`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 account-number or account-number-index to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AuthenticationText`
`config-property-type: java.lang.String`
`config-property-value:` This property specifies the last response message text from TIP Session Control on OS 2200 to indicate when a session has been successfully established. Default = "Previous session was:". Use "" if TIP Session Control is not configured on OS 2200.
- `config-property-name: LogFilename`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the directory path/file name of the OS 2200 Connector log file, where event and diagnostic trace information is recorded. If the path is a relative path name, the log file can be found in the `WAS_HOME\profiles\xxxxxx` directory of your WAS installation, where `WAS_HOME` is the WAS installation location and `xxxxxx` is the WAS profile name.
- `config-property-name: LoggingEnabled`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the event logging indicator. If true, event information is recorded in the OS 2200 Connector log file. The event log records primary events and errors in the usage of OS 2200 Connector.
- `config-property-name: Debug`
`config-property-type: java.lang.Boolean`

`config-property-value`: This property contains the OS 2200 Connector diagnostic trace indicator. If true, OS 2200 diagnostic trace entries are logged to the OS 2200 Connector log file.

- `config-property-name`: `MaximumCharacters`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property contains the maximum character String length for input and output data transported between OS 2200 Connector and the TIP/HVTIP transaction. Default = 512.
- `Config-property-name`: `InPort`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property contains the IP address that OS2200 Connector the resource adapter listens on for inbound requests. Default = 2498.
- `config-property-name`: `UseInboundSecureConnection`
`config-property-type`: `java.lang.Boolean`
`config-property-value`: This property contains the Secure Socket Layer (SSL) indicator for inbound communications. If true, the OS 2200 client program initiates an SSL handshake with OS 2200 Connector. An encryption method and a unique session key are established between the OS 2200 client program and OS 2200 Connector. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for inbound communications, the `InPort` `config-property` parameter must match a `CPComm` configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A private key and associated trusted digital certificate must be generated on the system containing the WebSphere Application Server Java Virtual Machine (JVM). The key must be imported into the Java system "keystore" and the certificate must be imported into the Java system "truststore." In the WAS startup script, the `javax.net.ssl.keyStore` Java system property must be set to the location of the "keystore"; that is, use the java option - `Djavax.net.ssl.keyStore=C:\...\my.keystore`. Also, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore." For more information on creating and managing keys, digital certificates, keystores and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation. Finally, the trusted certificate file must also be imported into the `trusted-certificates-file` declared in the OS 2200 ClearPath Server `CPComm` configuration.

- `config-property-name`: `RequestSocketTO`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that OS2200 Connector the resource adapter waits to receive more data from an inbound client request. A value of zero disables this feature. Default = 10000.
- `config-property-name`: `ServerSocketTO`
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that OS2200 Connector waits for a new inbound client request. A value of zero disables this feature. Default = 0.

The `ServerName`, `PortNumber`, `ConnectionTO`, `UseSecureConnection`, `UserName`, `Password`, `ClearanceLevel`, `ProjectId`, and `AccountNumber` parameters are used to define the default connection attributes for OS 2200 Connector. You may override these attributes in your

application component by creating an OS2200ConnectionSpec object with the new parameter values. Use the OS2200ConnectionSpec object to obtain the OS2200Connection object. For more information, see OS 2200 Connector API documentation, located at (assuming default installation path)

C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Unisys recommends that the remainder of the properties in ra.xml remain unchanged. Modify those properties only if necessary to tailor to your installation needs.

[Return to Top](#)

Related Topics

- [WebSphere Application Server, Version 8.0 Documentation - Configuring Resource Adapters](#)
- [WebSphere Application Server, Version 8.0 Information Center](#)
- [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#)
- [Using the WebSphere Application Server Administration Console](#)

Generating the Connector Archive--WAS

To create the OS 2200 Connector archive (RAR), follow these steps:

1. Create an empty staging directory.
2. Place the JAR containing the OS 2200 Connector Java classes (OS2200.jar) in the staging directory.
3. Place the deployment descriptor (ra.xml) in a subdirectory called META-INF.
4. Create the connector archive by executing a jar command like the following in the staging directory:

```
jar cvf OS2200.rar *
```

5. Verify the connector archive was built correctly by executing a jar command like the following:

```
jar -tf OS2200.rar
```

The OS2200.rar file structure should look like the following:

```
META-INF/
META-INF/MANIFEST.MF
META-INF/ra.xml
OS2200.jar
```

6. Deploy the RAR on WebSphere Application Server or include it in an enterprise archive (EAR) to be deployed as part of an enterprise application (See [Deploying the Connector--WAS](#)).

The OS 2200 Connector installation provides sample command scripts (setEnv.cmd and buildrar.cmd) that you can use, with site specific modifications, to build the RAR.

Related Topics

- [Deploying the Connector--WAS](#)
- [Using the WebSphere Application Server Administration Console](#)

Deploying the Connector--WAS

For detailed information on installing and deploying resource adapters on IBM WebSphere Application Server, see the [WebSphere Application Server, Version 8.0 Documentation - Installing J2EE Connector Resource Adapters](#) web page and the [WebSphere Application Server, Version 8.0 Documentation - Configuring J2EE Connector Connection Factories](#) web page.

To deploy OS 2200 Connector on WAS 8.0, follow these steps.

Note: WebLogic Application Server must be running and active.

Installing the Resource Adapter

1. Open the IBM WebSphere **First Steps** wizard.
2. Click **Start the server**.
3. After the server is started, open and log on to the WebSphere Application Server Administration Console: click **Administration console** or open `http://ip-address:9060/ibm/console` in your browser window, where *ip-address* is the WAS execution host location.
4. Expand the **Resources** entry in the left column.
5. Expand the **Resource Adapters** entry under **Resources**.
6. Select **Resource Adapters**.
7. Click **Install RAR**.
8. Under Local Path, browse to your OS 2200 RAR file in the staging directory used in generating the archive and select **OS2200.rar**. Click **Next**.
9. In the **Configuration** tab, if the Resource Adapter **Name** is not acceptable, enter a new name. Click **OK**.
10. In the **Messages** section, click **Save** to apply the changes to the WAS master configuration.

Unisys OS 2200 TIP Connector (default name), or the name you chose, now appears in the installed Resource Adapter list.

Configuring the OS 2200 Connector Connection Factory in WAS

1. Continuing with the screen in the previous steps, now click the connector name in the **Resource Adapter** list; in this case, **Unisys OS 2200 TIP Connector**.
2. Click **J2C connection factories** under the **Additional Properties** column.
3. To create a new connection factory, click **New**.
4. In the **General Properties** section of the **Configuration** tab, enter a name for the connection factory name, for example, **OS 2200 TIP Connection Factory**.

5. Enter a JNDI name for the connection factory, for example **eis/UNISYS_EIS_14_0/ConnectionFactory** (the JNDI name used by the sample EJBs supplied with the OS 2200 Connector installation).
6. Select **None for Authentication Preference**. Click **Apply**.
7. If you need to change connection pooling properties, click **Connection Pool Properties** under the **Additional Properties** column. Change the required properties and click **Apply**.
8. In the Messages section, click **Save** to apply the changes to the WAS master configuration.

The connector is now deployed but not started. WAS must be stopped and restarted. The WAS restart starts the connector and allows access to the connector from EJBs and other J2EE components.

Once deployed with this procedure, the deployed connector is automatically restarted when WAS restarts.

[Return to Top](#)

Related Topics

- [WebSphere Application Server, Version 8.0 Documentation - Installing J2EE Connector Resource Adapters](#)
- [WebSphere Application Server, Version 8.0 Documentation - Configuring J2EE Connector Connection Factories](#)
- [Generating the Connector Archive--WAS](#)
- [Editing the Deployment Descriptors--WAS](#)
- [Using the WebLogic Server Administration Console](#)

Connector Administration--WebSphere Application Server

Using the WebSphere Application Server Console

To administer OS 2200 Connector, use the WebSphere Application Server Administration Console.

For general information about using the WAS Administration Console, see the [WebSphere Application Server 8.0 Documentation--Administration Console](#) web page.

To open the WAS Administration Console, open a web browser window and browse to

`http://ip-address:9060/ibm/console`

where *ip-address* is the WAS execution host location.

Related Topic

[WebSphere Application Server 8.0 Documentation--Administration Console](#).

Editing the Deployment Descriptors--WAS

You can edit the ra.xml connector deployment descriptor using a text editor or XML editor.

You can edit the following OS 2200 Connector specific config properties of ra.xml:

- Debug
- LogFilename
- LoggingEnabled
- PortNumber
- ServerName
- ConnectionTO

- UseSecureConnection
- UserName
- Password
- ClearanceLevel
- ProjectId
- AccountNumber
- AuthenticationText

- MaximumCharacters
- InPort
- UseInboundSecureConnection
- RequestSocketTO
- ServerSocketTO

After you save the edited changes, delete OS 2200 Connector from the WAS instance and rebuild the RAR archive with the new .xml files. Then you can reinstall OS 2200 Connector on WAS.

Related Topics

- [Deploying the Connector -- WAS](#)
- [Using the WebSphere Application Server Administration Console](#)

Event Log and Diagnostic Trace Log--WAS

The J2EE Connector Architecture provides for a logging mechanism by which connectors can log arbitrary events to a predetermined file location. OS 2200 Connector uses this log mechanism to record primary events, warnings, errors, and diagnostic trace information during the execution of OS 2200 Connector. The log file is identified by the LogFilename property of the ra.xml deployment descriptor. Both the event log and the diagnostic trace log can be enabled and disabled independently using ra.xml properties.

OS 2200 Connector creates a log file set with a maximum of five cycles. The size limit for each log file in the set is 5MB. When the active file in the set reaches the 5MB size limit, this file is closed and the next file in the set is opened. When the last and fifth file in the set reaches the size limit, the oldest file in the set is purged.

OS 2200 Connector identifies the files in the set by appending a number, starting with 0, to the base file name. For example, if the base file name is OS2200Connector.log and the set consists of five files, the files are named OS2200Connector.log.0, OS2200Connector.log.1, OS2200Connector.log.2, OS2200Connector.log.3, and OS2200Connector.log.4. The OS2200Connector.log.0 file always contains the most recent cycle written. When OS 2200 Connector restarts, it appends log entries to the most recent log file cycle that was in use when OS 2200 Connector was stopped.

To view the log file, open it with a text editor.

Related Topics

- [Using the WebSphere Application Server Administration Console](#)
- [Configuring the Connector--WAS](#)

Using OS 2200 Connector with JBoss Application Server

Connector Installation and Configuration--JBoss Application Server

Installing the Connector--JBoss Enterprise Application Platform

Insert the OS 2200 Connector CD-ROM into the CD-ROM drive. Open setup.exe in the Win32 folder. Unisys recommends the default installation folder be named C:\Unisys\J2EE_Connector\OS2200 to coincide with the sample client and package generation scripts.

After installation, the OS 2200 Connector must be deployed as a component in the JBoss Enterprise Application Platform before any application components can use it. For information on OS 2200 Connector configuration and deployment, see [Configuring the Connector--JBoss Enterprise Application Platform](#).

Related Topics

- [Configuring the Connector--JBoss Enterprise Application Platform](#)
- [Using the JBoss Enterprise Application Platform Administration Console](#)

Configuring the Connector--JBoss Enterprise Application Platform

This section tells how to configure OS 2200 Connector for deployment to the JBoss Enterprise Application Platform.

OS 2200 Connector supports both JBoss Enterprise Application Platform 5.0 and JBoss Enterprise Application Platform (EAP) 6.0. Sections for configuring the connector for each platform are noted.

OS 2200 Connector must be deployed on JBoss EAP 5.0 or JBoss EAP 6.0. To learn about installing and administering JBoss, and to learn about deploying components within JBoss, see the [JBoss Enterprise Application Platform Product Documentation](#) web site.

After the OS 2200 Connector has been installed from the CD-ROM, a connector archive must be created and deployed. See [Generating the Connector Archive--JBoss Enterprise Application Platform](#).

ra.xml

Note: This section applies to both JBoss EAP 5.0 and JBoss EAP 6.0.

If you do not have an ra.xml deployment descriptor file, you must manually create or edit an existing one to set the necessary deployment properties for OS 2200 Connector. You can use a text editor to edit the properties. For information on the structure of the ra.xml deployment descriptor, see the [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#) web site.

OS 2200 Connector provides a sample ra.xml deployment descriptor file with its installation. You must edit the config-property sections of the ra.xml file to use your own OS 2200 Connector configuration properties.

- `config-property-name: ServerName`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the host name or IP address of the OS 2200 ClearPath server. If you are configuring OS 2200 Connector for use in a multi-host high availability environment with XTC (eXtended Transaction Capacity), this property will contain a semi-colon delimited list of configured host names or IP addresses.
- `config-property-name: PortNumber`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the port number of the OS 2200 Communications Application Program Interface listener defined in the CITA configuration. OS 2200 Connector opens a configuration dependent number of connections that are initially managed and pooled by JBoss Enterprise Application Platform upon OS 2200 Connector deployment.
- `config-property-name: ConnectionTO`
`config-property-type: java.lang.Integer`
`config-property-value:` This property specifies the connection timeout in milliseconds. This is the amount of time that an outbound client waits to receive a response or more data from an OS 2200 transaction. A value of zero disables this feature. Default = 0.
- `config-property-name: UseSecureConnection`
`config-property-type: java.lang.Boolean`
`config-property-value:` This property contains the Secure Socket Layer (SSL) indicator for outbound communications. If true, OS 2200 Connector initiates an SSL handshake with OS 2200 Communications Platform (CPComm). An encryption method and a unique session key are established between the OS 2200 Connector client and the OS 2200 ClearPath server. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for outbound communications, the PortNumber config-property parameter must match a CPComm configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A trusted certificate file that matches the certificate on the OS 2200 ClearPath Server

must be imported to a "truststore" on the system containing the JBoss Enterprise Application Platform Java Virtual Machine (JVM). In the JBoss startup script, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore"; that is, use the java option -

`Djavax.net.ssl.trustStore=C:\...\my.truststore`. For more information on creating and managing digital certificates and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation.

- `config-property-name: UserName`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 user-id to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: Password`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 password to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: ClearanceLevel`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 clearance-level to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: ProjectId`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 project-id or project-id-index to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AccountNumber`
`config-property-type: java.lang.String`
`config-property-value:` For the component-managed sign-on case, this property specifies the default OS 2200 account-number or account-number-index to be used for connection authentication with TIP Session Control on OS 2200. Default = "". Use default if TIP Session Control is not configured on OS 2200.
- `config-property-name: AuthenticationText`
`config-property-type: java.lang.String`
`config-property-value:` This property specifies the last response message text from TIP Session Control on OS 2200 to indicate when a session has been successfully established. Default = "Previous session was:". Use "" if TIP Session Control is not configured on OS 2200.
- `config-property-name: LogFilename`
`config-property-type: java.lang.String`
`config-property-value:` This property contains the directory path/file name of the OS 2200 Connector log file, where event and diagnostic trace information is recorded. If the path is a relative path name, the log file is in the bin directory of your JBoss Enterprise Application Platform installation.
- `config-property-name: LoggingEnabled`
`config-property-type: java.lang.Boolean`

`config-property-value`: This property contains the event logging indicator. If true, event information is recorded in the OS 2200 Connector log file. The event log records primary events and errors in the usage of OS 2200 Connector.

- `config-property-name`: Debug
`config-property-type`: `java.lang.Boolean`
`config-property-value`: This property contains the OS 2200 Connector diagnostic trace indicator. If true, OS 2200 diagnostic trace entries are logged to the OS 2200 Connector log file.
- `config-property-name`: MaximumCharacters
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property contains the maximum character string length for input and output data transported between OS 2200 Connector and the TIP/HVTIP transaction. Default = 512.
- `config-property-name`: InPort
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property contains the IP address that OS2200 Connector listens on for inbound requests. Default = 2498.
- `config-property-name`: UseInboundSecureConnection
`config-property-type`: `java.lang.Boolean`
`config-property-value`: This property contains the Secure Socket Layer (SSL) indicator for inbound communications. If true, the OS 2200 client program initiates an SSL handshake with OS 2200 Connector. An encryption method and a unique session key are established between the OS 2200 client program and OS 2200 Connector. The client and server begin a secure session that protects message privacy and message integrity. Default = false.

Note: To use the OS 2200 Connector SSL feature for inbound communications, the InPort config-property parameter must match a CPEndComm configured SSL port on the OS 2200 ClearPath Server. Refer to the *ClearPath Enterprise Servers Communications Platform Configurations and Operations Guide* for more details. A private key and associated trusted digital certificate must be generated on the system containing the JBoss Enterprise Application Platform Java Virtual Machine (JVM). The key must be imported into the Java system "keystore" and the certificate must be imported into the Java system "truststore." In the JBoss startup script, the `javax.net.ssl.keyStore` Java system property must be set to the location of the "keystore"; that is, use the java option `-Djavax.net.ssl.keyStore=C:\...\my.keystore`. Also, the `javax.net.ssl.trustStore` Java system property must be set to the location of the "truststore." For more information on creating and managing keys, digital certificates, keystores and truststores for Java, see *Sun Java Development Kit (JDK) keytool Security Tool* documentation. Finally, the trusted certificate file must also be imported into the *trusted-certificates-file* declared in the OS 2200 ClearPath Server CPEndComm configuration.

- `config-property-name`: RequestSocketTO
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that OS2200 Connector waits to receive more data from an inbound client request. A value of zero disables this feature. Default = 10000.
- `config-property-name`: ServerSocketTO
`config-property-type`: `java.lang.Integer`
`config-property-value`: This property specifies the time in milliseconds that

OS2200 Connector waits for a new inbound client request. A value of zero disables this feature. Default = 0.

The `ServerName`, `PortNumber`, `ConnectionTO`, `UseSecureConnection`, `UserName`, `Password`, `ClearanceLevel`, `ProjectId`, and `AccountNumber` parameters define the default connection attributes for OS 2200 Connector. To override these attributes in your application component, create an `OS2200ConnectionSpec` object with the new parameter values. Use the `OS2200ConnectionSpec` object to obtain the `OS2200Connection` object. For more information, see OS 2200 Connector API documentation, located at (assuming default installation path) `C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html`. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Unisys recommends that the remainder of the properties in `ra.xml` remain unchanged. Modify those properties only if necessary to tailor to your installation needs.

[Return to Top](#)

OS2200-ds.xml

Note: This section applies to JBoss EAP 5.0.

In addition to supporting features of the standard connector configuration `ra.xml` deployment descriptor file, JBoss Enterprise Application Platform requires an additional deployment descriptor file, a *datasourcename-ds.xml* file. This file contains properties that are specific to configuring and deploying connectors and data sources in JBoss Enterprise Application Platform. OS 2200 Connector provides a sample `OS2200-ds.xml` deployment descriptor file with its installation.

Edit the following properties of the `OS2200-ds.xml` file to suit your needs:

- `jndi-name` property, to configure your own JNDI name.
- `min-pool-size` property and `max-pool-size` property, for connection pooling and management.
- `application-managed-security` property, for managing connection pools using component-managed sign-on.
- `security-domain` property, for defining application policies using container-managed sign-on.

`OS2200-ds.xml` is not part of the `OS2200.rar` deployment package used to deploy OS 2200 Connector in JBoss Enterprise Application Platform; however, it must exist in the same deployment directory ("all" or "default") as the `OS2200.rar` file. See [Deploying the Connector--JBoss Enterprise Application Platform](#).

[Return to Top](#)

jboss-deployment-structure.xml and module.xml

Note: This section applies to JBoss EAP 6.0.

In addition to supporting features of the standard connector configuration ra.xml deployment descriptor file, JBoss EAP requires additional deployment configuration files, a jboss-deployment-structure.xml and module.xml files. These files contain properties that are specific to configuring and deploying connectors in JBoss EAP. OS 2200 Connector provides a sample jboss-deployment-structure.xml and module.xml files with its installation.

There is no need to manually edit these files.

jboss-deployment-structure.xml is part of the OS2200.rar deployment package used to deploy OS 2200 Connector in JBoss EAP. module.xml is not part of the OS2200.rar deployment but must exist in the JBoss MODULE directory. See [Deploying the Connector--JBoss Enterprise Application Platform](#).

[Return to Top](#)

Related Topics

- [Java Platform, Enterprise Edition \(Java EE\) - Connector Architecture Downloads & Specifications](#)
- [Deploying the Connector--JBoss Enterprise Application Platform](#)
- [Generating the Connector Archive--JBoss Enterprise Application Platform](#)
- [Using the JBoss Enterprise Application Platform Administration Console](#)

Generating the Connector Archive--JBoss Enterprise Application Platform

This section tells how to generate the OS 2200 Connector archive for JBoss Enterprise Application Platform.

OS 2200 Connector supports both JBoss Enterprise Application Platform 5.0 and JBoss Enterprise Application Platform (EAP) 6.0. Sections for generating the connector archive for each platform are noted.

JBoss Enterprise Application Platform 5.0

To create the OS 2200 Connector archive (RAR) for JBoss EAP 5.0, follow these steps:

1. Create an empty staging directory.
2. Place the JAR containing the OS 2200 Connector Java classes (OS2200.jar) in the staging directory.
3. Place the deployment descriptor (ra.xml) in a subdirectory called META-INF.
4. Create the connector archive by executing a jar command like the following in the staging directory:

```
jar cvf OS2200.rar *
```

5. Verify the connector archive was built correctly by executing a jar command like the following:

```
jar -tf OS2200.rar
```

The OS2200.rar file structure should look like the following:

```
META-INF /
META-INF/MANIFEST.MF
META-INF/ra.xml
```

6. Deploy the RAR on JBoss Enterprise Application Platform (in the "all" or "default" server configuration) or include it in an enterprise archive (EAR) to be deployed as part of an enterprise application. See [Deploying the Connector--JBoss Enterprise Application Platform](#).

The OS 2200 Connector installation provides sample command scripts (setEnv.cmd and buildrar.cmd) that you can use, with site-specific modifications, to build the RAR.

[Return to Top](#)

JBoss Enterprise Application Platform 6.0

To create the OS 2200 Connector archive (RAR) For JBoss EAP 6.0, follow these steps:

1. Create an empty staging directory.
2. Place the JAR containing the OS 2200 Connector Java classes (OS2200.jar) in the staging directory.
3. Place the deployment descriptors (ra.xml and jboss-deployment-structure.xml) in a subdirectory called META-INF.
4. Create the connector archive by executing a jar command like the following in the staging directory:

```
jar cvf OS2200.rar *
```

5. Verify the connector archive was built correctly by executing a jar command like the following:

```
jar -tf OS2200.rar
```

The OS2200.rar file structure should look like the following:

```
META-INF /
META-INF/MANIFEST.MF
META-INF/ra.xml
META-INF/jboss-deployment-structure.xml
```

6. Deploy the RAR on JBoss EAP (in the "standalone" or "domain" server configuration) or include it in an enterprise archive (EAR) to be deployed as part of an enterprise application. See [Deploying the Connector--JBoss Enterprise Application Platform](#).

The OS 2200 Connector installation provides sample command scripts (setEnvJB6.cmd and buildrarJB6.cmd) that you can use, with site-specific modifications, to build the RAR.

[Return to Top](#)

Related Topics

- [Deploying the Connector--JBoss Enterprise Application Platform](#)
- [Using the JBoss Enterprise Application Platform Administration Consoles](#)

Deploying the Connector--JBoss Enterprise Application Platform

This section tells how to deploy OS 2200 Connector to JBoss Enterprise Application Platform.

OS 2200 Connector supports both JBoss Enterprise Application Platform 5.0 and JBoss Enterprise Application Platform (EAP) 6.0. Sections for configuring the connector for each platform are noted.

JBoss Enterprise Application Platform 5.0

To deploy OS 2200 Connector on JBoss EAP 5.0, copy the OS2200.rar file and the OS2200-ds.xml file to a deployment directory within JBoss Enterprise Application Platform. The standard deployment directories are

JBOSS_HOME\server\all\deploy for the "all" server configuration or
JBOSS_HOME\server\default\deploy for the "default" server configuration.

Also copy the OS2200.jar file to the \lib directory within your JBoss server configuration. The standard lib directories are:

JBOSS_HOME\server\all\lib for the "all" server configuration or
JBOSS_HOME\server\default\lib for the "default" server configuration.

"JBOSS_HOME" represents the JBoss Enterprise Application Platform installation location.

Once deployed with this procedure, the connector is automatically redeployed when JBoss Enterprise Application Platform restarts.

The OS 2200 Connector installation provides sample command scripts (setEnv.cmd and buildrar.cmd) that you can use, with site-specific modifications, to deploy the OS 2200 Connector deployment package.

[Return to Top](#)

JBoss Enterprise Application Platform 6.0

To deploy OS 2200 Connector on JBoss EAP 6.0, follow these steps:

Note: JBoss EAP must be running and active.

Installing the Resource Adapter

1. Start the server.
2. Open and log on to the JBoss Management Console. Open `http://ip-address:9990/console` in your browser window, where *ip-address* is the JBoss execution host location.
3. Click the **Runtime** tab at the top right of the Management Console.
4. Select **Manage Deployments** under **Server** in the left navigation window.
5. Click the **Add Content** button in the **Deployments** panel.
6. In the **Upload/Deployment Selection** dialog box, browse to your OS 2200 RAR or EAR file and select it. Click **Next**.
7. The **Deployments** panel shows the connector module marked in the **Disabled** state.

Configuring the OS 2200 Connection Factory

1. Click the **Profile** tab at the top right of the Management Console.
2. Select **Resource Adapters** under **Connector** in the left navigation window.
3. Click the **Add** button in the **JCA Resource Adapters** panel.
4. In the **Create Resource Adapter** dialog box, enter your OS 2200 Connector archive name (OS2200.rar) in the **Archive** text box, and select **NoTransaction** in the **TX** list. Click **Save**.
5. The **JCA Resource Adapters** panel shows an OS2200.rar entry for **Available Resource Adapter**. Click **View**.
6. Here we define the OS 2200 Connector connection factory. In the **Available Connection Definitions** panel, click the **Add** button.
7. In the **Create Connection Definition/Connection Definition Step 1** dialog box, for the **JNDI Name**, enter, for example, **java:/eis/UNISYS_EIS_14_0/ConnectionFactory** (the JNDI name used by the sample EJBs supplied with the OS 2200 Connector installation). For the **Connection Class**, enter **com.unisys.os2200.connector.OS2200ManagedConnectionFactory**. Click **Next**.
8. In the **Create Connection Definition/Connection Definition Step 2** dialog box, click **Save**.
9. If you need to change connection pooling properties, click **Pool** tab in the **Available Connection Definitions** panel. Click the **Edit** button.
10. Change the required property values and click **Save**.
11. The **Available Connection Definitions** panel shows the updated property values.
12. The **Available Connection Definitions** panel shows the connector module marked in the **Disabled** state. Click **Enable**.
13. In the **Modify Connection Definition** dialog box, click **Confirm**.
14. The **Available Connection Definitions** panel now shows the connector module marked in the **Enabled** state.

Reloading the Server

1. The top of the Management Console shows an information box which says "A server reload is required!". Click the box.
2. In the **Message** dialog box, click **OK**.
3. Click the **Runtime** tab at the top right of the Management Console.
4. Select **Configuration** under **Server** in the left navigation window.

5. In the **Server Configuration** panel, click the **Reload** button.
6. In the **Reload server configuration** dialog box, Click **Confirm**.
7. The message "The server configuration is up to date!" appears in the **Server Configuration** panel.

Enabling the Resource Adapter

1. Click the **Runtime** tab at the top right of the Management Console.
2. Select **Manage Deployments** under **Server** in the left navigation window.
3. The **Deployments** panel shows the connector module marked in the **Disabled** state. Click **Enable**.
4. In the **Are you sure?/Enable OS2200.rar?** dialog box, click **Confirm**.
5. The **Deployments** panel shows the connector module marked in the **Enabled** state.

Once deployed and enabled with this procedure, the connector is automatically redeployed when JBoss EAP restarts.

[Return to Top](#)

Related Topics

- [Generating the Connector Archive--JBoss Enterprise Application Platform](#)
- [Editing the Deployment Descriptors--JBoss Enterprise Application Platform](#)
- [Using the JBoss Enterprise Application Platform Administration Consoles](#)

Connector Administration--JBoss Application Server

Using the JBoss Enterprise Application Platform Administration Consoles

JBoss Enterprise Application Platform 5.0

The JBoss EAP 5.0 JMX Console Web Application (referred to as the JMX Console) provides a raw view into the JBoss EAP microkernel and allows you to manage several features and functions of your application server.

To view detailed properties of OS 2200 Connector, use the JBoss JMX Console. To open the console, open a Web browser window and browse to

```
http://ip-address:8080/jmx-console
```

where *ip-address* is the JBoss Enterprise Application Platform execution host location.

JBoss Enterprise Application Platform 6.0

The JBoss EAP Management Console 6.0 (referred to as the Management Console) allows you to manage several features and functions of your application server. You can use the Management Console to deploy and undeploy applications, configure and deploy resource adapters, tune JBoss system settings, and make persistent modifications to the server configuration. The Management Console also provides live notifications when any changes require the server instance to be restarted or reloaded.

To open the console, open a Web browser window and browse to

`http://ip-address:9990/console`

where *ip-address* is the JBoss EAP execution host location.

Editing the Deployment Descriptors--JBoss Enterprise Application Platform

This section tells how to edit OS 2200 Connector deployment descriptors for JBoss Enterprise Application Platform.

OS 2200 Connector supports both JBoss Enterprise Application Platform 5.0 and JBoss Enterprise Application Platform (EAP) 6.0. Sections for editing the deployment descriptors for each platform are noted.

JBoss Enterprise Application Platform 5.0

You can edit the following connector deployment descriptors using a text editor or XML editor:

- ra.xml
- OS2200-ds.xml

You can edit the following OS 2200 Connector specific config properties of ra.xml:

- Debug
- LogFilename
- LoggingEnabled
- PortNumber
- ServerName
- ConnectionTO
- UseSecureConnection
- UserName
- Password
- ClearanceLevel
- ProjectId
- AccountNumber
- AuthenticationText
- MaximumCharacters
- InPort
- UseInboundSecureConnection
- RequestSocketTO
- ServerSocketTO

You can edit the following properties of OS2200-ds.xml:

- jndi-name
- min-pool-size
- max-pool-size

- security-domain

After you save the edited changes, rebuild the RAR archive with the new ra.xml file. Then you can redeploy OS 2200 Connector without stopping JBoss Enterprise Application Platform.

If you redeploy a new or updated version of the OS2200.jar file, you must restart JBoss Enterprise Application Platform for the changes to take effect.

[Return to Top](#)

JBoss Enterprise Application Platform 6.0

You can edit the ra.xml connector deployment descriptor using a text editor or XML editor.

You can edit the following OS 2200 Connector specific config properties of ra.xml:

- Debug
- LogFilename
- LoggingEnabled
- PortNumber
- ServerName
- ConnectionTO
- UseSecureConnection
- Username
- Password
- ClearanceLevel
- ProjectId
- AccountNumber
- AuthenticationText
- MaximumCharacters
- InPort
- UseInboundSecureConnection
- RequestSocketTO
- ServerSocketTO

After you save the edited changes, remove OS 2200 Connector from the JBoss EAP instance using the JBoss EAP Management Console. Rebuild the RAR archive with the new .xml file, and redeploy OS 2200 Connector on JBoss EAP using the Management Console.

[Return to Top](#)

Related Topics

- [Deploying the Connector--JBoss Enterprise Application Platform](#)
- [Using the JBoss Enterprise Application Platform Administration Consoles](#)

Event Log and Diagnostic Trace Log--JBoss Enterprise Application Platform

The J2EE Connector Architecture provides a logging mechanism by which connectors can log arbitrary events to a predetermined file location. OS 2200 Connector uses this log

mechanism to record primary events, warnings, errors, and diagnostic trace information during the execution of OS 2200 Connector. The log file is identified by the LogFilename property of the ra.xml deployment descriptor. Both the event log and the diagnostic trace log can be enabled and disabled independently using ra.xml properties.

OS 2200 Connector creates a log file set with a maximum of five cycles. The size limit for each log file in the set is 5MB. When the active file in the set reaches the 5MB size limit, this file is closed and the next file in the set is opened. When the last and fifth file in the set reaches the size limit, the oldest file in the set is purged.

OS 2200 Connector identifies the files in the set by appending a number, starting with 0, to the base file name. For example, if the base file name is OS2200Connector.log and the set consists of five files, the files are named OS2200Connector.log.0, OS2200Connector.log.1, OS2200Connector.log.2, OS2200Connector.log.3, and OS2200Connector.log.4. The OS2200Connector.log.0 file always contains the most recent cycle written. When OS 2200 Connector restarts, it appends log entries to the most recent log file cycle that was in use when OS 2200 Connector was stopped.

To view the log file, open it with a text editor.

Related Topics

- [Using the JBoss Enterprise Application Platform Administration Console](#)
- [Configuring the Connector \(ra.xml\)--JBoss Enterprise Application Platform](#)

Developing Your Solutions

Overview

The development process usually involves two kinds of tasks:

Client development tasks

- Outbound communications: The development tasks can be performed on any Java platform where suitable development tools for the client and EJB involved are installed. The deployment of the EJB must be performed on the same computer on which Oracle WebLogic Server, IBM WebSphere Application Server, or JBoss Enterprise Application Platform is running.
- Inbound communications: The client development tasks are performed on the OS 2200 ClearPath host partition. The client programs can be developed using the COBOL, C, or Java language.

Server/Service development tasks

- Outbound communications: These tasks are usually performed on the OS 2200 ClearPath host partition.
- Inbound communications: These tasks are performed on the Java application server platform in the form of Enterprise Java Beans or Plain Old Java Objects (POJO).

Related Topics

- [WebLogic Server EJB and Client Development](#)
- [WebSphere Application Server EJB and Client Development](#)
- [JBoss Enterprise Application Platform EJB and Client Development](#)
- [Inbound Communications](#)
- [Outbound Communications](#)
- [EJB/POJO Development for Inbound Communications](#)

Applications Using Communications Application Program Interface or Message Control Bank

Outbound Communications

- [Developing a Communications Application Program Interface or MCB Application](#)
- [Using an Existing MCB Application or DPS Application](#)
- [Sample MCB and Communications Application Program Interface Programs](#)

Host development tasks vary according to the details of your implementation. If you are targeting an existing and unmodified OS 2200 application for use with OS 2200 Connector, the target application must use Message Control Bank (MCB) for message handling. DPS 2200 (Display Processing System) target applications are also supported, if using MCB for message handling. If you are creating a new application or modifying an existing application to use with OS 2200 Connector, you can use either MCB or Communications Application Program Interface for message handling.

MCB is the message control component of the OS 2200 integrated recovery system. MCB provides additional functions, particularly in message recovery. It is used primarily in a transaction processing environment. For more information, see the *Message Control Bank (MCB) Programming Reference Manual*.

DPS 2200 is an application development tool that defines device independent TIP display screens through the use of its development-time utilities and run-time execution. Use the DPS FLMU utility with the CONNECTOR working storage option to generate a COBOL formed record that can be used as input and output data record structures. For more information, see the *Display Processing System (DPS 2200) Forms Design Programming Guide*.

Communications Application Program Interface is a combined extended-mode subsystem and background run that allows applications access to a communications network. It provides a sockets-like interface for application programs using either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connections.

Communications Application Program Interface offers improved data transfer efficiency by bypassing much of the OS 2200 software that normally provides integrated data recovery. This greatly reduces the internal path length for simple transactions that may not require full data recovery capability. If necessary, an application that uses Communications Application Program Interface can be designed to provide some form of data recovery. For more information, see the *Communications Application Program Interface User's Guide*.

Developing a Communications Application Program Interface or MCB Application

To create and network-enable a Communications Application Program Interface or MCB application for use with OS 2200 Connector, perform the following steps:

Step One: Define the Input and Output Message Data Structures

If applicable, define the required message data structures for the Communications Application Program Interface or MCB application. The RecordName attributes (or VIEWS) identify the structure of the data used in the transfer of information between the Java platform and the OS 2200 ClearPath server. A Java application component uses the OS2200Record data object to pass the RecordName to the Communications Application Program Interface or MCB application. Due to the different data representation formats in the two operating environments, the data passed is limited to ASCII characters only.

Note: The RecordName field may be left blank. A blank indicates that the resulting data is unformatted as it is passed on requests.

Step Two: Create a Communications Application Program Interface or MCB Application

To create a Communications Application Program Interface or MCB application on the OS 2200 ClearPath server, you can both design and code a new application, or you can modify an existing application. The Communications Application Program Interface or MCB application must follow these rules:

- Transactions must use MCB for input message handling. Transaction message output can either be MCB or Communications Application Program Interface. The Communications Interface for Transaction Applications (CITA) dispatcher used with Communications Application Program Interface interfaces with MCB.
- Depending on the CITA configuration, the transaction input and output data may optionally include a OS 2200 Connector-specific data header, followed by the data definition optionally defined by a RecordName. The CITA configuration sample and related Readme.txt provided with the OS 2200 Connector installation illustrates a configuration entry that satisfies this requirement. For more information on configuring CITA, see the *Communications Interface for Transaction Applications Configuration and Operations Guide*.

The OS 2200 Connector-specific data header contains the following:

- A 6-byte character field to indicate the OS 2200 transaction code
- A 1-byte character field to indicate the header descriptor version
- A 6-byte character field to indicate the total data length being sent--this includes the length of the OS 2200 Connector-specific data header along with the length of the RecordName data
- A 16-byte character field (space-filled) to indicate the RecordName in the RecordName description
- A 3-byte character filler

The following C structure shows a data definition that could be used for the OS 2200 Connector-specific data in a Communications Application Program Interface or MCB application:

```
struct tipra{
    char trancode[6];
    char tipra_version;
    char byte_length[6];
    char record_name[16];
    char filler[3];
};
```

The following C structure shows a RecordName data definition that could be used in a Communications Application Program Interface or MCB application:

```
struct EMP_BUF {
    char lastname[30];
    char firstname[30];
    char mi;
    char empno[6];
    char ssno[11];
    char annivdate[9];
    char plant[5];
};
```

The following COBOL structure shows a data definition that could be used for the OS 2200 Connector-specific data in a Communications Application Program Interface or MCB application:

```
01  TIPRA-HEADER-BUFFER.
    05  TRANCODE                PIC X(6).
    05  TXI-VERSION             PIC X(1).
    05  BYTE-LENGTH             PIC X(6).
    05  RECORD-NAME             PIC X(16).
    05  FILLER                  PIC X(3).
```

[Return to Top](#)

Using an Existing MCB or DPS Application

To use an existing and unmodified MCB or DPS application with OS 2200 Connector, the MCB application must follow this rule: transactions must use MCB for both input and output message handling.

The CITA dispatcher interfaces with MCB. You must configure the CITA product to eliminate the passing of the OS 2200 Connector-specific header data to the application. You must also configure CITA to properly handle the OS 2200 authentication dialogue, if TIP Session Control is configured on the ClearPath server. The CITA configuration sample provided with the OS 2200 Connector installation illustrates configuration entries that satisfy these requirements. See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\OS2200Tools, for a description of the CITA configuration sample. For more information on configuring CITA, see the *Communications Interface for Transaction Applications Configuration and Operations Guide*.

Sample MCB and Communications Application Program Interface Programs

The OS 2200 Connector installation provides a sample C application (adempl.c) and a sample COBOL Compiler (previously called UCOB) application (aducob.ucob), which use MCB, the Communications Application Program Interface, and a RecordName description for both input and output data. The installation also provides an example of a legacy COBOL Compiler application (legacy.ucob), which uses only MCB for both input and output data. Also included are sample OS 2200 ECL runstreams to compile and link the example transaction applications. Use FTP or CIFS to transfer the sample applications and runstreams to the host OS 2200 ClearPath server for compilation and execution.

For more information on creating Communications Application Program Interface applications, see the *Communications Application Program Interface User's Guide*.

For more information on creating MCB applications, see the *Message Control Bank (MCB) Programming Reference Manual*.

Once your sample program has been successfully compiled and linked, the program must be registered with TIP. The program needs a VALTAB associated with it. You may have to consult with the system administrator to have a VALTAB assigned and registered with TIP.

Once a program has been registered with TIP, a port number must be assigned to one or more of the sample programs. The OS 2200 communications product, Communications Interface for Transaction Applications (CITA), listens on the assigned port number. For more information on assigning a port to the sample programs, see the *Communications Interface for Transaction Applications Configuration and Operations Guide*. The OS 2200 Connector installation provides a sample CITA configuration (cita.config), which sets up the interface to the network and allows the sample transactions to be accessible to the OS 2200 Connector client/EJB.

See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\OS2200Tools, for instructions on how to build and set up the sample C and COBOL transactions on the OS 2200 ClearPath server.

[Return to Top](#)

Inbound Communications

Developing Inbound Calls Using Communications Application Program Interface

Use the Communications Application Program Interface to make OS 2200 Connector inbound calls from a COBOL or C application by creating the TCP socket connections.

Communications Application Program Interface is a combined extended mode subsystem and background run that allows applications to access a communications network. It provides a sockets-like interface for application programs using either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connections.

Sequence of Communications Application Program Interface Calls

OS 2200 applications that use the OS 2200 Connector inbound feature must make several required Communications Application Program Interface calls. The following table outlines the sequence of these calls.

Call or Processing Step	Description
Call to TCP_SESSION	The application calls the Communications Application Program Interface command TCP_SESSION to create a session table. Once the session table has been created, the application is now ready to make a socket connection.
Loading the LRCT Data Structure	<p>Before the application can make a call to the Communications Application Program Interface command TCP_CONNECT, which makes the socket connection, the LRCT data structure must be loaded. The LRCT is a buffer declared for you in the following element or file:</p> <ul style="list-style-type: none"> • COBOL: The COBOL COPY PROC COMAPI-UCOB (SYS\$LIB\$*COMAPI.COMAPI-DEF) • C: The include file SYS\$LIB\$*COMAPI.APIDEF/H <p>The LRCT structure contains fields that identify a version, the communication interface that Communications Application Program Interface is using (either CMS 1100 or Communications Platform), the local IP address and port number, and the remote IP address and port number. The LRCT must contain the IP address and port number of the remote peer application. If you specify a valid value in the LRCT interface field, it overrides the interface selection in the session table.</p>
Call to TCP_CONNECT	Once the LRCT is properly loaded, the application can call the TCP_CONNECT command.
Loading the Header and Data for TCP_SEND	<p>Once the connection has been made, the application should load the 72-byte OS 2200 Connector inbound header, EJB name, and data into a continuous data buffer to be sent using the Communications Application Program Interface command TCP_SEND. The OS 2200 Connector inbound header is a buffer declared for you in the OS 2200 Connector installation locations:</p> <ul style="list-style-type: none"> • COBOL: The COBOL COPY PROC slheader.cobp in C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client\ • C: The include file slheader.h in C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client
Set Blocking with	After the data has been sent, it is recommended to set the

TCP_SETOPTS	connection to a blocking of type 2. This allows OS 2200 Connector to read a specified number of bytes using the java.net and java.io packages. To do this, use the TCP_SETOPTS command.
Calls to TCP_RECEIVE	Once the data has been sent, the target EJB responds with a data message. OS 2200 Connector prefixes the data with the 72-byte OS 2200 Connector inbound header. Two TCP_RECEIVE calls should be made: one call to read the 72-byte header; and once the total message length is determined by the Byte-Length header field, a second TCP_RECEIVE call to receive the rest of the data.
Call to TCP_DISCONNECT	Close the socket connection.

See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client, for instructions on how to build and set up the sample C and COBOL transactions on the OS 2200 ClearPath server.

[Return to Top](#)

Developing Inbound Calls Using Message Control Bank (MCB)

Use the MCB to make OS 2200 Connector inbound calls from a COBOL or C application by creating the MCB outbound opens.

MCB is the message control component of the OS 2200 integrated recovery system. MCB provides additional functions, particularly in message recovery. It is used primarily in a transaction processing environment. For more information, see the *Message Control Bank (MCB) Programming Reference Manual*.

The nature of MCB outbound opens does not allow for a conversational model for the inbound clients. In other words, the same transaction cannot send the EJB request and receive the EJB response. There must be two separate transactions, one sends data and the other receives the response.

Sequence of MCB Calls

OS 2200 applications that use the OS 2200 Connector inbound feature must make several required MCB calls. The following table outlines the sequence of these calls.

Call or Processing Step	Description
Execute with V Option	The batch or TIP application must be executed with the V option. This allows the application to open MCB outbound sessions.
MCB TRINIT\$\$	Initialize with MCB; read an input message if necessary.

MCB OPNSSF\$	Open an outbound session for a PID. The PID must be defined in the CITA configuration. The CITA configuration sample provided with the OS 2200 Connector installation illustrates a configuration entry that satisfies these requirements. See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\OS2200Tools, for a description of the CITA configuration sample. For more information on configuring CITA, see the <i>Communications Interface for Transaction Applications Configuration and Operations Guide</i> .
MCB SEND\$	Once the outbound session is open, the application should load the 72-byte OS 2200 Connector inbound header, EJB name, and data into a continuous data buffer to be sent using the MCB SEND\$. The OS 2200 Connector inbound header is a buffer declared for you in the OS 2200 Connector installation locations: <ul style="list-style-type: none"> • COBOL: The COBOL COPY PROC slheader.cobp in C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client\ • C: The include file slheader.h in C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client
MCB TERM\$	Terminate with MCB.
A second TIP transaction receives EJB response	With a session already open, the target EJB sends a response to a second TIP transaction. CITA is involved with the transaction's scheduling, using the same PID bound to the MCB outbound open. The CITA configuration sample provided with the OS 2200 Connector installation illustrates a configuration entry that satisfies these requirements. See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\OS2200Tools, for a description of the CITA configuration sample. For more information on configuring CITA, see the <i>Communications Interface for Transaction Applications Configuration and Operations Guide</i> .
MCB TRINIT\$	Initialize with MCB; the transaction reads the input message. The input message may or may not contain the 72-byte OS 2200 Connector inbound header, depending on the CITA configuration. The CITA configuration sample provided with the OS 2200 Connector installation eliminates the header from the data before delivery to the TIP transaction.
MCB TERM\$	Terminate with MCB.

Note: OS 2200 Connector closes the MCB session after the EJB sends its response.

See the Readme.txt file in the default installation folder, C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\Client, for instructions on how to build and set up the sample COBOL batch program and transaction on the OS 2200 ClearPath server.

[Return to Top](#)

Developing Inbound Calls Using Java

Use the **java.net** and **java.io** packages to make OS 2200 Connector inbound calls from a Java application by creating the TCP socket connections.

Use the OS2200HeaderRev3 class to create the 72-byte OS 2200 Connector inbound header. The message header has the following format:

PacketID	(4 chars)
Interface-Level	(1 char)
Status	(3 chars)
Conversation-Mode	(1 char)
Filler	(3 chars)
Input-Type	(4 chars)
Output-Type	(4 chars)
Bean-Type	(4 chars)
BeanName-Length	(8 chars)
Byte-Length	(10 chars)
Filler	(30 chars)

For more detail on the fields and access methods in OS2200HeaderRev3 class, consult the OS 2200 Connector API documentation, located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

For the *Interface-Level* field, the current value is "3".

For the *Status* field, OS 2200 Connector reserves error codes "000" through "099" for internal error conditions passed to the client application. The EJB application may use error codes "100" through "999".

OS 2200 Connector generated error codes for the *Status* field are:

- 0 Good return, no errors.
- 1 Invalid *PacketID* field.
- 2 Invalid *Interface-Level* field.
- 3 Invalid *Input-Type* field.
- 4 Invalid *Output-Type* field.
- 5 Invalid *Bean-Type* field.

- 6 `com.unisys.os2200.connector.OS2200HeaderException` thrown. Unable to parse OS2200Header for request message.
- 7 `javax.ejb.EJBException` thrown. EJBException thrown by target EJB.
- 8 `java.io.InterruptedIOException` thrown. Socket connection to client has timed out.
- 9 `java.net.SocketException` thrown. Socket connection to the client has failed.
- 10 `java.io.IOException` thrown. Socket connection to the client has failed.
- 11 `javax.ejb.CreateException` thrown. Unable to create an instance of the target EJB.
- 12 `javax.naming.NamingException` thrown. Unable to find the target EJB.
- 13 `java.rmi.RemoteException` thrown. Unable to send data to the target EJB.
- 14 `java.lang.ClassCastException` thrown. ClassCastException thrown in target EJB lookup.
- 15 `com.unisys.os2200.connector.OS2200HeaderException` thrown. Unable to create response header for return from EJB.
- 16 `java.lang.Exception` thrown (generic).

Related Topic

- [Enterprise JavaBeans and Client Development](#)

Enterprise JavaBeans and Client Development

WebLogic Server EJB and Client Development

The OS 2200 Connector installation provides sample EJB 2.0 and EJB 3.0 beans, and test clients, which you can verify in a two- or three-tier architecture system. The EJB running in the Oracle WebLogic Server container invokes the Common Client Interface (CCI) provided by OS 2200 Connector. The CCI provides the interaction with the OS 2200 ClearPath server. For a list of classes, see OS 2200 Connector API documentation, located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

See the Readme.txt file in the default installation folder C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\WebLogic\EJB2 for instructions on how build and execute the sample EJB 2.0 and test client for WebLogic Server.

See the Readme.txt file in the default installation folder
 C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\WebLogic\EJB3 for instructions on how to build and execute the sample EJB 3.0 and test client for WebLogic Server.

WebSphere Application Server EJB and Client Development

The OS 2200 Connector installation provides sample EJB 2.0 and EJB 3.0 beans and test clients, which you can verify in a two- or three-tier architecture system. The EJB running in the WebSphere Application Server container invokes the Common Client Interface (CCI) provided by OS 2200 Connector. The CCI provides the interaction with the OS 2200 ClearPath server. For a list of classes, see OS 2200 Connector API documentation, located at (assuming default installation path)
 C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

See the Readme.txt file in the default installation folder
 C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\WebSphere\EJB2 for instructions on how to build and execute the sample EJB 2.0 and test client for WebSphere Application Server.

See the Readme.txt file in the default installation folder
 C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\WebSphere\EJB3 for instructions on how to build and execute the sample EJB 3.0 and test client for WebSphere Application Server.

JBoss Enterprise Application Platform EJB and Client Development

The OS 2200 Connector installation provides sample EJB 2.0 and EJB 3.0 beans and test clients, which you can verify in a two- or three-tier architecture system. The EJB running in the JBoss Enterprise Application Platform container invokes the Common Client Interface (CCI) provided by OS 2200 Connector. The CCI provides the interaction with the OS 2200 ClearPath server. For a list of classes, see OS 2200 Connector API documentation, located at (assuming default installation path)
 C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

See the Readme.txt file in the default installation folder
 C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\JBoss\EJB2 for instructions on how to build and execute the sample EJB 2.0 and test client for JBoss Enterprise Application Platform.

See the Readme.txt file in the default installation folder
 C:\Unisys\J2EE_Connector\OS2200\ExampleOutbound\JBoss\EJB3 for instructions on how to build and execute the sample EJB 3.0 and test client for JBoss Enterprise Application Platform.

EJB/POJO Development for Inbound Communications

EJB 2.0 Development

Any EJB 2.0 or EJB 2.1 EJB written for use with OS 2200 Connector must use predefined interfaces. The following interfaces are located in the `com.unisys.os2200.connector` package found in the `OS2200.jar` file.

- `OS2200EjbRemoteHome`: The remote home interface for inbound-compliant EJBs (EJB 2.0).
- `OS2200EjbRemoteObject`: The remote object interface for inbound-compliant EJBs (EJB 2.0).

Note: During development of your inbound-compliant EJBs, add the `OS2200.jar` file to your classpath. Also, import the `com.unisys.os2200.connector` package into your EJB source code.

For more information and sample EJBs for each supported application server, see the examples and the `Readme.txt` file in the samples directories (default:

`C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebLogic\EJB2`,
`C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebSphere\EJB2`, and
`C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\EJB2`).

The `OS2200EjbRemoteHome` Interface

This is the remote home interface for inbound-compliant remote EJBs (EJB 2.0). This interface extends the `javax.ejb.EJBHome` interface. All inbound-compliant EJBs must use `OS2200EjbRemoteHome` as their remote home interface.

The `OS2200EjbRemoteObject` Interface

This is the remote object interface for inbound-compliant remote EJBs (EJB 2.0). This interface extends the `javax.ejb.EJBObject` interface. All inbound-compliant EJBs must use `OS2200EjbRemoteObject` as their remote interface. The `os2200EjbService()` method that is defined in the `OS2200EjbRemoteObject` interface is the only method that can be called by OS 2200 Connector inbound.

The `OS2200EjbReturnData` Class

Contains the return data that is sent back from calls to the `OS2200EjbRemoteObject` `os2200EjbService()` method that is implemented in OS 2200 Connector inbound communications-compliant EJBs. This data is returned to OS 2200 Connector as a subclass of the `OS2200Record` object (`OS2200Request` or `OS2200Xoctet`) and then returned to the client over the socket connection. The `retCode` and `retRecord` parameters that are included in the `OS2200EjbReturnData` object are used to hold the error status and return data that are sent back to the client application. The following parameters are set in the constructors for `OS2200EjbReturnData`.

- *int* `retCode`: Holds the system-defined or application-defined return code.
- *OS2200Record* `retRecord`: Holds the application-defined return data.

OS 2200 Connector inbound is not concerned with the data that is contained in these parameters. This data only has meaning to the client application and target EJB. The data is simply passed from the EJB to the client.

The `retCode` can be found in the `Status` field of `OS2200Header` class, which is received by the client application. For the `Status` field, OS 2200 Connector reserves error codes "000" through "099" for internal error conditions. The EJB application may use error codes "100" through "999".

For more detail on the `Status` field for the `OS2200Header` class, consult the OS 2200 Connector API documentation, located at (assuming default installation path) `C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html`. It is also accessible from **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

The `OS2200EjbSessionBase` Class

`OS2200EjbSessionBase` is a convenience class for creating session beans for OS 2200 Connector inbound communications. `OS2200EjbSessionBase` contains empty implementations of the methods specified by the `javax.ejb.SessionBean` interface. Session beans created for use with the inbound communications need only to extend this class and contain implementations for the `os2200EjbService()` and `ejbCreate()` methods.

Configuring EJBs (EJB 2.0) for Inbound Communication

`ejb-jar.xml`

This is the main configuration file for EJBs and it is mandatory that your EJB package contains an `ejb-jar.xml` configuration file. This file should be placed in the `/META-INF` directory of each `.jar` file that contains inbound-compliant EJBs. For more information on the `ejb-jar.xml` configuration file, see the Sun J2EE EJB 2.0 Specification. The following tags must have the specified values:

<code><ejb-name></code>	The reference name of the EJB. This is also the default JNDI name.
<code><home></code>	This value must be set to <code>com.unisys.os2200.connector.OS2200EjbRemoteHome</code>
<code><remote></code>	This value must be set to <code>com.unisys.os2200.connector.OS2200EjbRemoteObject</code>
<code><ejb-class></code>	Specifies the implementation class of the EJB.

Specifying the JNDI Names

EJBs are referenced by their JNDI names for inbound communications. This name is passed to the OS 2200 Connector from the client in the OS 2200 Connector inbound header. Each supported application server has its own unique method for defining specific JNDI names. In the absence of a specified JNDI name, the value of the `<ejb-name>` tag in the `ejb-jar.xml` configuration file is used as the JNDI name. For example, if `<ejb-name>` is set to "Foo", then the EJB is accessed with the name JNDI "Foo."

The JNDI naming behavior can be modified using configuration mechanisms specific for each application server. See the application server specific documentation for information on how to set the JNDI name. The `Readme.txt` files in the inbound communication samples directories also contain information about setting the JNDI name (default: `C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebLogic\EJB2`,

C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebSphere\EJB2, and
 C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\EJB2).

EJB 3.0 Development

Any EJB 3.0 EJB written for use with OS 2200 Connector must use a predefined interface. The following interface is located in the `com.unisys.os2200.connector` package found in the `OS2200.jar` file.

- `OS2200Ejb3RemoteObject`: The remote object interface for inbound-compliant EJBs (EJB 3.0).

Note: During development of your inbound-compliant EJBs, add the `OS2200.jar` file to your classpath. Also, import the `com.unisys.os2200.connector` package into your EJB source code.

For more information and sample EJBs for each supported application server, see the examples and the `Readme.txt` file in the samples directories (default:
 C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebLogic\EJB3,
 C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebSphere\EJB3, and
 C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\EJB3).

The `OS2200Ejb3RemoteObject` Interface

This is the remote object interface for inbound-compliant remote EJBs (EJB 3.0). This interface makes use of the `@Remote` annotation. All inbound-compliant EJBs must use `OS2200Ejb3RemoteObject` as their remote interface. The `os2200EjbService()` method that is defined in the `OS2200Ejb3RemoteObject` interface is the only method that can be called by OS 2200 Connector inbound.

The `OS2200EjbReturnData` Class

Contains the return data that is sent back from calls to the `OS2200Ejb3RemoteObject` `os2200EjbService()` method that is implemented in OS 2200 Connector inbound communications-compliant EJBs. This data is returned to OS 2200 Connector as a subclass of the `OS2200Record` object (`OS2200Request` or `OS2200Xoctet`) and then returned to the client over the socket connection. The `retCode` and `retRecord` parameters that are included in the `OS2200EjbReturnData` object are used to hold the error status and return data that is sent back to the client application. The following parameters are set in the constructors for `OS2200EjbReturnData`.

- `int retCode`: Holds the system-defined or application-defined return code.
- `OS2200Record retRecord`: Holds the application-defined return data.

OS 2200 Connector inbound is not concerned with the data that is contained in these parameters. This data only has meaning to the client application and target EJB. The data is simply passed from the EJB to the client.

The `retCode` can be found in the `Status` field of `OS2200Header` class, which is received by the client application. For the `Status` field, OS 2200 Connector reserves error codes "000"

through "099" for internal error conditions. The EJB application may use error codes "100" through "999".

For more detail on the *Status* field for the OS2200Header class, consult the OS 2200 Connector API documentation, located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Specifying the JNDI Names

EJBs are referenced by their JNDI names for inbound communications. This name is passed to the OS 2200 Connector from the client in the OS 2200 Connector inbound header.

For EJB 3.0, each supported application server (WLS, WAS, and JBoss Enterprise Application Platform) has its own unique method for defining default JNDI names.

For example, for JBoss, the value of the EJB implementation class name followed by "/remote" is used as the JNDI name. If the EJB implementation class name is set to "Foo", then the EJB is accessed with the JNDI name "Foo/remote."

The Readme.txt files in the inbound communication sample directory also contain information about setting the default EJB 3.0 JNDI names in each supported application server (default: C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebLogic\EJB3, C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\WebSphere\EJB3, and C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\EJB3).

Spring POJO Bean Development

This section is JBoss Enterprise Application Platform specific.

Any Spring POJO (Plain Old Java Object) Bean written for use with OS 2200 Connector must use a predefined interface. The following interface is located in the com.unisys.os2200.connector package found in the OS2200.jar file.

- OS2200PojoRemoteObject: The remote object interface for inbound-compliant Spring POJO Beans).

Note: During development of your inbound-compliant POJOs, add the OS2200.jar file to your classpath. Also, import the com.unisys.os2200.connector package into your POJO source code.

Note: The Spring Framework library package (spring.jar or could be multiple .jar's) must be added to your classpath. For more information on Spring Framework, see the [Spring Framework](#) web site.

For more information and sample POJOs for JBoss Enterprise Application Platform, see the examples and the Readme.txt file in the samples directory (default: C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\POJO).

The OS2200PojoRemoteObject Interface

This is the remote object interface for inbound-compliant remote Spring POJO beans. All inbound-compliant Spring POJO beans must use `OS2200PojoRemoteObject` as their remote interface. The `os2200EjbService()` method that is defined in the `OS2200PojoRemoteObject` interface is the only method that can be called by OS 2200 Connector inbound.

The `OS2200EjbReturnData` Class

Contains the return data that is sent back from calls to the `OS2200PojoRemoteObject` `os2200EjbService()` method that is implemented in OS 2200 Connector inbound communications-compliant Spring POJO beans. This data is returned to OS 2200 Connector as a subclass of the `OS2200Record` object (`OS2200Request` or `OS2200Xoctet`) and then returned to the client over the socket connection. The `retCode` and `retRecord` parameters that are included in the `OS2200EjbReturnData` object are used to hold the error status and return data that is sent back to the client application. The following parameters are set in the constructors for `OS2200EjbReturnData`.

- `int retCode`: Holds the system-defined or application-defined return code.
- `OS2200Record retRecord`: Holds the application-defined return data.

OS 2200 Connector inbound is not concerned with the data that is contained in these parameters. This data only has meaning to the client application and target POJO bean. The data is simply passed from the POJO bean to the client.

The `retCode` can be found in the `Status` field of `OS2200Header` class, which is received by the client application. For the `Status` field, OS 2200 Connector reserves error codes "000" through "099" for internal error conditions. The Spring POJO bean application may use error codes "100" through "999".

For more detail on the `Status` field for the `OS2200Header` class, consult the OS 2200 Connector API documentation, located at (assuming default installation path) `C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html`. It is also accessible from **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

Configuring Spring POJO Beans for Inbound Communication

`*-spring.xml`

This is the JBoss configuration file for Spring POJO beans and it is mandatory that your Spring POJO package contains a `*-spring.xml` configuration file. This file should be placed in the root directory of each `.jar` package that contains inbound-compliant Spring POJO beans. For more information on the `*-spring.xml` configuration file, see the [JBoss Snowdrop Project](#) web site. The Spring POJO bean sample provided with the OS 2200 Connector installation includes a `*-spring.xml` file named `tipra-jboss-spring.xml`. For more information on configuring the sample `tipra-jboss-spring.xml` file, see the `Readme.txt` file in Spring POJO sample directory (default: `C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\POJO`).

JNDIExporter Class

The `JNDIExporter` class object is a required implementation of the Spring Framework `org.springframework.beans.factory.DisposableBean` and

org.springframework.beans.factory.InitializingBean interfaces. This implementation is the Spring JNDI invoker bean and must be included in the Spring POJO bean package.

JBoss Spring Deployer

The JBoss Spring Deployer MBean must be deployed as a component of the configured JBoss server instance. For more information on downloading the JBoss Spring Deployer, see the [JBoss Snowdrop Project](#) web site.

Spring Framework Library

The Spring Framework library package (spring.jar or could be multiple .jar's) must be added to the "\lib" directory of the configured JBoss server instance. For more information on Spring Framework, see the [Spring Framework](#) web site.

Specifying the JNDI Names

Spring POJO beans are referenced by their JNDI names for inbound communications. This name is passed to the OS 2200 Connector from the client in the OS 2200 Connector inbound header. For Spring POJO beans, the value of the *jndiName* property in the *-spring.xml configuration file is used as the JNDI name. For example, if *jndiName* property is set to "Foo", then the Spring POJO bean is accessed with the JNDI name "Foo."

The Readme.txt files in the inbound communication sample directory also contain information about setting the JNDI name for Spring POJOs (default:
C:\Unisys\J2EE_Connector\OS2200\ExampleInbound\EJB\JBoss\POJO).

Related Topics

- [Spring Framework](#)
- [JBoss Snowdrop Project](#)

Client Development in a Nonmanaged Environment

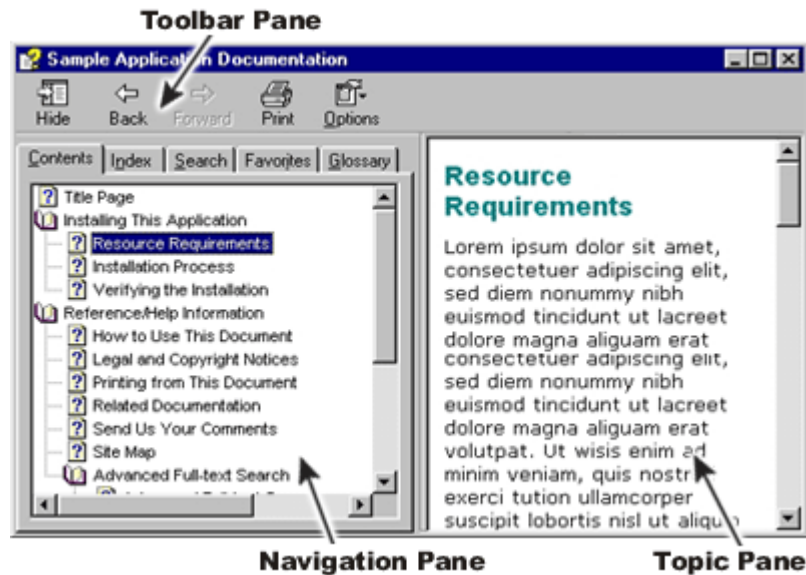
The OS 2200 Connector installation provides a sample test client, which you can verify in a two-tier nonmanaged system. The client interacts directly with OS 2200 Connector rather than interacting with an application server such as WLS, WAS, or JBoss Enterprise Application Platform. The test client invokes the Common Client Interface (CCI) provided by OS 2200 Connector. The CCI provides the interaction with the OS 2200 ClearPath server. For a list of classes, see OS 2200 Connector API documentation, located at (assuming default installation path) C:\Unisys\J2EE_Connector\OS2200\DocumentationAPI\index.html. It is also accessible from the **Start > Programs > Unisys OS 2200 TIP Connector > API Documentation**.

See the Readme.txt file in the default installation folder
C:\Unisys\J2EE_Connector\OS2200\ExampleNonManaged for instructions on how build and execute the test client for a nonmanaged environment.

Reference/Help Information

How to Use This HTML Help Document

The document is typically displayed in a window that includes some or all of the following panes:




Note: Some topics displayed in secondary windows will differ from the default window shown above. Secondary windows include only the features that are useful for a particular type of topic. A secondary window may not include a navigation pane, or the toolbar pane may contain a different selection of buttons.











The following are explanations of the panes in the default window:

- [Using the toolbar pane](#)
- [Using the navigation pane](#)
- [Using the topic pane](#)
- [Restoring link color and resizing or hiding panes](#)

Using the Toolbar Pane

The toolbar pane can display any of the following buttons:





Toolbar Buttons	Description
	<p>Enables you to redisplay pages that you have previously viewed.</p> <ul style="list-style-type: none"> • The Back button returns you to the last page you viewed before opening the current page. If you use the Back button repeatedly, it takes you in reverse order through the sequence of pages you have

 Forward	<p>already viewed.</p> <ul style="list-style-type: none"> The Forward button takes you forward through a sequence of pages that you previously viewed. <p>Contrast these buttons with the Previous and Next buttons.</p>
 Hide  Show	<p>Hides the navigation pane. When the navigation pane is hidden, clicking the Show button restores the pane.</p>
 Home	<p>Displays the title page. This is the page that is initially displayed when the document opens.</p>
 Locate	<p>Updates the table of contents to show which topic is currently open.</p>
 Options	<p>Offers a number of selections:</p> <ul style="list-style-type: none"> Back, Forward, Home, Locate, Refresh, Stop, and Print. These selections serve the same purpose as the toolbar buttons of the same name. These selections are available even if the corresponding buttons do not appear on the toolbar. Hide Tabs and Show Tabs. Hides or shows the navigation pane. Internet Options. Displays a number of options for the Internet Explorer browser. Search Highlight On/Off. Highlights every found instance of a searched word.
 Print	<p>Prints one or more topics. A dialog box prompts you to choose between</p> <ul style="list-style-type: none"> Printing the current topic. Printing all the topics in the book currently selected in the Contents tab. <p>See Printing from This Document for additional information.</p>
 Refresh	<p>Reloads the current topic from the source file.</p>
 Site Map	<p>Displays a file that shows all of the topics in the help project, normally grouped by subject. (Optional)</p>
 Stop	<p>Stops the loading of the current topic.</p>
<p>External</p>	<p>Goes to an external Web site or link (for example, the Unisys Support Site).</p>

Jump	
------	--

Using the Navigation Pane




The navigation pane can contain any of the following tabs:

Tab Name	Description
Contents	<p>Presents a hierarchical, expandable/collapsible table of contents. You can display each subject area in as much or little detail as you want. The following symbols are used in the table of contents:</p> <ul style="list-style-type: none">  A "book" with subtopics that are not currently displayed. Click the book to display the subtopics.  A "book" with subtopics that are already displayed. Click the book to hide the subtopics.  A topic that has no subtopics beneath it. Click to display the topic.  A link to a document on the World Wide Web. Click to display the document.
Index	<p>A list of index entries. Type a term you are looking for in the Type in the word(s) to search for box, or scroll through the list until you find a term that interests you. Click the term in the list to display the topic.</p>
Search	<p>Provides a full text search.</p> <ol style="list-style-type: none"> 1. Type one or more search words in the Type in the word(s) to search for box. 2. Click List Topics. 3. Scroll through the Topics list, and double-click the topic that interests you.
Favorites	<p>Displays a list of favorite topics, along with the following buttons</p> <ul style="list-style-type: none"> • To add the current topic to the favorites list, click Add. • To display a topic from the favorites list, double-click the name of the topic. • To remove a topic from the favorites list, select the topic and then click Remove. • To rename a topic in the favorites list, click the topic name, pause, and then click the topic name

	again. Type the new name and press Enter .
Glossary	A list of terms and definitions. In the Term box, select a term. The definition appears in the Definition for box.

Using the Topic Pane

The topic pane displays the text of the current help topic. Following are the meanings of some icons that you might see in help topics:

Icons	Description
 Software Caution	Indicates information about preventing possible damage to equipment or loss of data.
 Hardware Caution	
 WARNING	Indicates information about preventing injuries or fatalities.

Restoring Link Color and Resizing or Hiding Panes

Action	Procedure
To restore links to their original color	<p>To clear your visited links so they appear in their original (unvisited) color, take the following steps:</p> <ol style="list-style-type: none"> 1. Click Options on the toolbar. 2. Click Internet Options. <p>The Internet Options dialog box appears.</p> <ol style="list-style-type: none"> 3. Click Clear History. 4. Click Yes when the confirmation message appears. 5. Click OK in the Internet Options dialog box. 6. Click Refresh on the toolbar. If there is no Refresh button, click Options and then click Refresh on the menu. <p>Note: This process clears all visited links in the Internet Explorer history, not only the links in this document.</p>
To resize a pane	Drag the border to where you want it to be.

To hide or restore the navigation pane	Click Hide or Show on the toolbar pane.
--	---

Legal and Copyright Notices

Copyright © 2013 Unisys Corporation

All rights reserved.

ClearPath Enterprise Servers OS 2200 Transaction Resource Adapter for the Java™ Platform User's Guide

4729 1992-007 (associated with ClearPath OS 2200 Release 14.0 or subsequent releases)

Warranty Disclaimer

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Notice to U.S. Government End Users: This is commercial computer software or hardware documentation developed at private expense. Use, reproduction, or disclosure by the Government is subject to the terms of Unisys standard commercial license for the products, and where applicable, the restricted/limited rights provisions of the contract data rights clauses.

Third-Party Web Sites

Unisys Corporation does not make any representations regarding any third-party Web site that might be linked to this document. Such linked Web sites are independent of Unisys Web sites and Unisys does not have any control over the content and/or use of any such site. Such links are provided only as a convenience, and the inclusion of any link to such sites does not imply endorsement by Unisys of those sites.

Trademarks

Unisys and ClearPath are registered trademarks of [Unisys Corporation](#) in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

JBoss is a registered trademark of Red Hat, Inc. in the United States and other countries.

All other brands and products referenced in this document are acknowledged to be the trademarks or registered trademarks of their respective holders.

Other Copyright Notices

Portions copyright © [Adobe Systems Incorporated](#). All rights reserved.

Portions copyright © [Microsoft Corporation](#). All rights reserved. 2012-12-07 15:04:18

Printing From This HTML Help Document

You can print sections of this document for offline use. The printed version contains the information in the online version but the formatting, fonts, and style sheets might be different from the online view.

Printing a Topic Displayed in the Topic Pane

Click the **Print** icon on the toolbar pane, select **Print the selected topic**, and click **OK**. Or right-click the topic displayed in the topic pane and click **Print**.

Printing a Section of This Document

Right-click a section heading in the table of contents, and click **Print**. Then, select **Print the selected heading and all subtopics**, and click **OK**. The HTML Help Viewer prints all the topics in the current section, and all topics in subordinate sections, but not necessarily in order.

Notes:

- The header of each page contains the title of the first topic in the section.
- Because of a problem with the Microsoft HTML Help Viewer when used with Internet Explorer 5.5 (and later) and some Windows environments, occasionally the last word in a line of text might be truncated or repeated on the next line. Print each topic individually to avoid this problem.

Printing This Entire Document

Right-click the top-level book, and click **Print**. Then, select **Print the selected heading and all subtopics**, and click **OK**. The HTML Help Viewer prints all the topics in this document, but not necessarily in order. Each page header contains the document title.

Notes:

- If a window entitled "? HTML Help" comes up before the **Print** dialog box, ignore the "? HTML Help" window. The "? HTML Help" window contains the title page. When the **Print** dialog box comes up and you click **Print** or **Cancel**, the "? HTML Help" window

disappears. A bug in the Microsoft HTML Help Viewer or Internet Explorer causes this error.

- Because of a problem with the Microsoft HTML Help Viewer when used with Internet Explorer 5.5 (and later) and some Windows environments, occasionally the last word in a line of text might be truncated or repeated on the next line. Print each topic individually to avoid this problem.

Index

A

administration 16, 24, 35
applications 48
 Enterprise JavaBeans 47, 48
archive generation 14, 22, 31

B

BEA WebLogic..... 10, 14, 15, 16, 17, 47

C

client & EJB development 47, 48
configuration 10, 18, 26
copyright notices..... 58

D

deployment 15, 16, 23, 24, 32, 35
descriptors--deployment 16, 24, 35
diagnostic trace 17, 25, 37

E

EJB & client development 47, 48
Enterprise JavaBeans..... 47, 48
event log..... 17, 25, 37

H

help on how to use this HTML Help
 document 54
how to use this HTML Help document ... 54

I

IBM WebSphere Application Server 18, 22,
 23, 24, 25, 47
installation 10, 18, 26
introduction..... 2, 4, 38
 developing solutions with OS 2200
 Connector 38
 Java EE Connector Architecture 2
 OS 2200 Connector 4

J

Java EE connector architecture 2
JBoss Enterprise Application Platform... 26,
 31, 32, 35, 37, 48

L

legal notices 58

N

new features 1

O

overview 2, 4, 38
 developing solutions with OS 2200
 Connector 38
 Java EE Connector Architecture 2
 OS 2200 Connector 4

P

printing from this HTML Help document 59

T

title page for this document	1
trace--diagnostic	17, 25, 37
trademarks	58

W

WebSphere Application Server .	18, 22, 23, 24, 25, 47
what's new?	1